Risk-taking and Recovery in Task-Oriented Dialogue

Jean Carletta

Ph.D.
University of Edinburgh
1992

I declare that this thesis has been composed by myself and that the work described in it is my own.

(Jean C. Carletta)

Acknowledgements

Many thanks to Chris Mellish for supervising this thesis with his usual patience, insight, and good humour. This would have been a much less interesting thesis without his guidance.

Thanks to Steve Isard for being my second supervisor and especially for helping with the analysis given in the third and fourth chapters of the thesis.

Thanks to Alison Cawsey for many words of wisdom (and even more cups of tea), especially in the early stages of the thesis.

Thanks to the AI department's natural language group and the discourse workshop at Cognitive Science for many useful discussions about both my own work and related topics. Thanks especially to Diana Bental, Paul Brna, Robert Dale, Judy Delin, Jacqueline Kowtko, Ian Lewin, Graeme Ritchie, and Peter Ross for occasional but very useful comments about my work. Thanks also to Alan Black for setting up the Lisp environment which I used when I wrote the JAM system.

Thanks to Steve Gifford for proofreading the entire thesis even if he does think that making computers talk is a silly thing to do.

Finally, thanks to the Marshall Aid Commemoration Commission for the generous financial support which made this thesis possible.

Abstract

The Principle of Parsimony states that by and large, agents try to complete tasks using as little effort as possible. This thesis demonstrates that the Principle of Parsimony operates in human task-oriented dialogue by showing the effects of Parsimony in a corpus of human dialogues about a map navigation task and by using the main points of the analysis in order to guide simulated conversations between two computer agents within the JAM system. It makes four major contributions:

- an analysis of "communicative posture", or a range of choices in dialogue which can be characterised by decisions about how much effort to spend constructing one's utterances, leading to either careful or risky behaviour about different aspects of communication,
- an analysis of "recovery strategies" which allow the participants to recover from failures which have been brought about due to risky postures,
- a heuristic model of belief which risks failing to capture the full meaning of the dialogue in favour of efficiency in a way which simulates human belief updating more plausibly than previous models, and
- a layered agent architecture which allows the simulated agents to make all of their decisions based on the Principle of Parsimony.

Contents

1 Introduction 1

1.1 The Approach: Corpus Analysis Plus Implementation 2

1.2 The Major Contributions 5

1.3 The Corpus Analysis 5

1.3.1 First Contribution: An Analysis of "Communicative Posture" 6

1.3.2 Second Contribution: An Analysis of Recovery 7

1.4 The JAM System 8

1.4.1 Third Contribution: The Belief Model 8

1.4.2 Fourth Contribution: The System Architecture 9

1.5 The Organisation of the Thesis 10

2 Review of Related Work 11

2.1 Modelling an Agent's Choices 12

2.1.1 Hovy's PAULINE 12

2.1.2 Shadbolt's Communicative Postures 13

2.2 Dialogue 15

2.2.1 Dialogue Games 16

2.2.2 Turn-taking 18

2.2.3 Simulating Both Agents in a Dialogue 19

2.3 Issues in Implementing Dialogue Systems 20
2.3.1 Interleaving Planning and Execution 20
2.3.2 Characterising an Agent's Goals 21
2.3.3 Meta-Planning 22
2.4 Recovery Strategies 25
2.4.1 Classifying Misunderstandings 25
2.4.2 Past Approaches to Recovery 26
2.4.3 Using Recovery in a Program 27
2.5 Belief Modelling 29
2.5.1 Theorem Proving Approaches 29
2.5.2 Degrees of Acquaintance 31
2.6 Summary 32

3 Communicative Posture **34**

3.1 Extending Shadbolt's Analysis 34
3.1.1 Parameters for the Explainee 35
3.1.2 Grouping the Parameters by Planning Stage 36
3.1.3 Caveats about "Success" and "Risk" 36
3.1.4 Methodological Issues 37
3.2 The Parameters 38
3.2.1 Task Planning Parameters 39
3.2.2 Discourse Planning Parameters 46
3.2.3 Utterance Realisation Parameters 49
3.2.4 Meta-Planning Parameter: Plan Commitment 57
3.3 Extension to Other Domains 58
3.4 Summary 61

4 The Recovery Strategies **62**

4.1 Communicative Recovery Strategies 63

4.1.1 Goal Adoption 64

4.1.2 Ceding Control to the Partner 65

4.1.3 Elaboration 66

4.1.4 Omission 68

4.2 General Purpose Strategies 70

4.2.1 Repetition 70

4.2.2 Ignoring the Problem 71

4.2.3 Repair 71

4.2.4 Replanning 73

4.2.5 Reinstantiation 75

4.3 Extension to Other Domains 76

4.4 Summary 78

5 The JAM System **79**

5.1 The Domain Knowledge Representation 80

5.2 The Map Databases 82

5.3 Goals 82

5.4 Communication and Dialogue Games 83

5.5 Plans 87

5.6 The Planning Language 90

5.6.1 Constraints 91

5.6.2 Prerequisites 92

5.7 The Planning Operators 93

5.8 How the Plan Operators are Used 96

5.9 How Current Plan Costs are Maintained 98

5.10 The Top Level Control Structure 99

5.11 The Control for an Agent 100

5.12 How Communicative Posture is Implemented 101

5.12.1 Implementing the Task Planning Parameters 101

5.12.2 Implementing the Difference Parameter 105

5.12.3 Implementing the Context Articulation Parameter 106

5.12.4 Implementing the Plan Commitment Parameter 107

5.13 Summary 107

6 The Belief Model **108**

6.1 Degrees of Acquaintance 110

6.2 The Interface with the System 112

6.2.1 The Interface in the Abstract 112

6.2.2 The Implemented Interface 114

6.3 An Abstract Description of the Belief Model 115

6.4 An Example Belief Representation 116

6.4.1 Exploiting the Regularities of Dialogue 116

6.4.2 Exploiting the Structure of the Domain 119

6.4.3 The Agent's Algorithm upon Awakening 121

6.4.4 A Worked Example 124

6.5 Modelling Uncertainty 128

6.5.1 Explicitly Representing only Certain States 128

6.5.2 Explicitly Representing Uncertain States 130

6.6 Combining the Network with a Theorem Prover 132

6.7 The Belief Representation for the JAM System 134

6.8 Conclusions 136

7 Recovering from Plan Failure using a Layered Architecture 139

7.1 Where Moore's Approach Fails 140

7.2 How the Strategies are Implemented 142

7.2.1 Communicating 143

7.2.2 Planning 144

7.2.3 Replanning 144

7.2.4 Repair 146

7.2.5 Goal Adoption 147

7.3 Estimating the Cost of a Continuation 148

7.4 The Layered Architecture 151

7.4.1 The Organisation of the Layers 152

7.4.2 The Domain Space 154

7.4.3 The Strategy Space 155

7.4.4 The Mode Space 156

7.4.5 The Interpreter 157

7.5 Worked Examples 157

7.5.1 Communication, Planning, Repair, and Goal Adoption 157

7.5.2 Replanning 164

7.6 Conclusions 166

8 Output from the JAM System 168

8.1 A Low Risk Dialogue 169

8.2 A High Risk Dialogue Without Recovery 171

8.3 Repair by the Route Follower 172

8.4 Repair by the Route Giver 176

8.5 Replanning 177
8.6 The Influence of Plan Commitment 178
8.7 A Dialogue Initiated by the Route Follower 180
8.8 Dialogues JAM Can't Generate 181
8.9 Conclusions 183

9 Conclusions and Further Work **185**

9.1 Further Work on Empirical Evidence 186
9.2 Further Work on Communicative Posture 187
9.3 Further Work on Recovery Strategies 188
9.4 Further Work on the Dialogue Interaction 189
9.5 Further Work on the Model of Belief 190
9.6 Further Work on the Layered Architecture 191
9.7 Other Further Work 193
9.8 Conclusions 193

A The Maps **196**

B Space Calculations **210**

C The Belief Model for the Simulation **213**

D How the Belief Model is Implemented **216**

D.1 How Belief Updating is Implemented 216
D.2 How Queries about Beliefs are Handled 218

Chapter 1

Introduction

This thesis is concerned with what people say when they have to cooperate to solve a particular task, that of describing a route on a map. There are a great number of decisions which agents must make in order to engage in task-oriented dialogue. The kinds of decisions which we will be concerned with in this thesis are directly related to the agents' primary goal of getting across information about the route: how to describe the route, how to organise the description, how to decide upon a series of utterances which will convey the description, and how to realise each of those utterances in language. We hypothesise that there is one principle, the Principle of Parsimony, which governs the decisions which we discuss: by and large, people try to complete tasks with as little effort as possible [Sha84]. Thus people will make whatever decisions about what to say in the dialogue they believe will bring about completion of the task using the least effort. Part of this thesis describes a computer program, JAM, which allows two computer agents to complete the map task using AI planning techniques to implement a model of the decision making observed in a corpus of human map task dialogues. In dialogue, unlike in more traditional planning domains, it is quite often more efficient to behave carelessly, and take the risk of having to recover from failures in one's plans, than to behave carefully in the first place. This technique is frequently used in the human dialogues. This leads us to the main theme of our thesis: that one of the things which makes human dialogue seem more natural than traditional computer dialogue is that the Principle of Parsimony leads human agents to run into plan failures and then use strategies for recovering from these failures, whereas computer dialogue agents rely on more careful behaviour in the first place. Thus one way to make computer dialogue seem more natural is to make it use this same strategy of failure and recovery that humans use, by giving the computer agent some notion of the effort involved in an utterance and using Parsimony to govern its behaviour.

1.1 The Approach: Corpus Analysis Plus Implementation

Our approach to supporting our claim is to show that humans use this strategy and then to demonstrate that the kinds of decision making and recovery from failure required to simulate their behaviour can be incorporated into a working system. We will do this by taking as given without further discussion the naive, one line version of the Principle of Parsimony which has already been presented and demonstrating that it can be used to explain much of the variety of behaviour which occurs in the map task dialogues. Thus, in the same vein as much A.I. work, we will not justify our main assumption or claim that our analysis is psychologically plausible in any meaningful way, but only that it "works" and is a useful way of thinking about task-oriented dialogue, both as an explanation of human behaviour and for the purposes of building computer dialogue agents. This thesis can be divided into two parts: an analysis of human task-oriented dialogues, and a system description. The task domain which we will be using throughout this thesis is one which was first described in [BASY84], in which two participants who are separated by a partition have slightly different versions of a simple map with approximately fifteen gross features on it, such as the two versions of the same map which occur in figures 1.1 and 1.2. The maps may have different features or have some of the features in different locations. In addition, one participant, the "route giver" or agent "A", has a route drawn on the map. The task is for the second participant, the "route follower" or agent "B", to duplicate the route. The agents are given instructions that indicate that the two versions of the maps are "drawn by different explorers", and therefore may vary somewhat.[1] This task provides the opportunity for a great many different dialogue decisions, failures, and recoveries but closely constrains the topic of the dialogue.

A fundamental part of our approach is that we provide a working simulation of some of the dialogue decisions, failures, and recoveries which are discussed in the analysis. The JAM system provides important support for the analysis, since it gives concrete examples of dialogue agents behaving according to the principles uncovered in the analysis and allows their behaviour to be compared with the original data. Implementation helps to highlight parts of an analysis which are underspecified or ambiguous. Experimentation with the system gives a sense of some of the strengths and weaknesses of the analysis which may not be obvious without a computational model. Finally, having a working system which uses the principles of the analysis in order to generate dialogue which seems more natural than traditional computer dialogue supports our

[1] These instructions are reproduced in appendix A along with the maps to which we will refer in the remainder of the thesis.

Figure 1.1: The Route Giver's Map

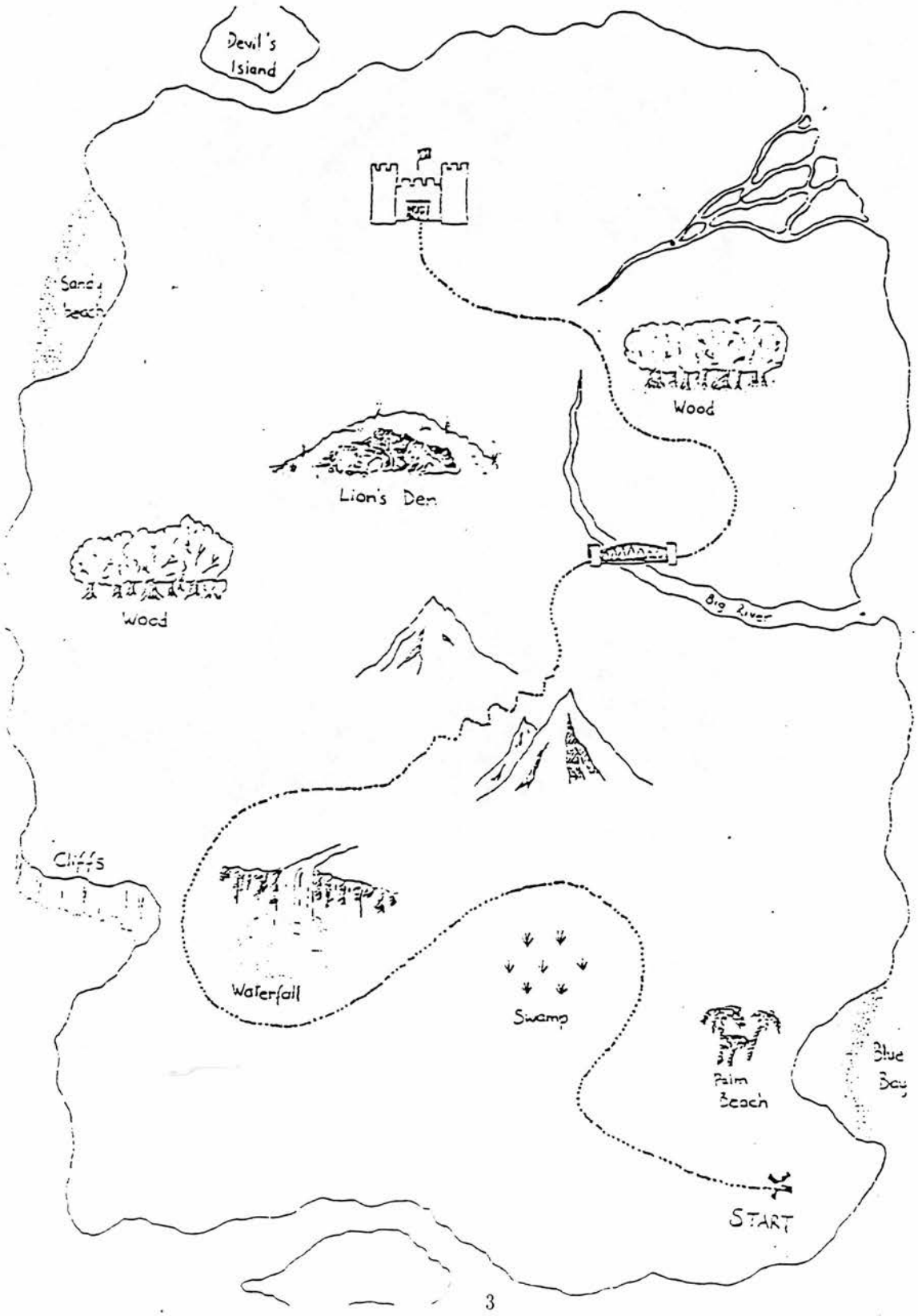


Figure 1.2: The Route Follower's Map

claim that failure and recovery are essential to human dialogue and should be considered in future computer dialogue systems.

1.2 The Major Contributions

This thesis makes four major contributions: two related to the analysis of a human corpus of task-oriented dialogues and two related to the simulation of this data using the JAM computer system. These contributions are:

- an analysis of some kinds of decisions which agents make during dialogue (called "communicative posture"), which can be characterised by a choice either to risk completion of some part of the task on the current attempt or to expend extra effort in order to safeguard success;
- an analysis of the strategies which agents use to recover from failures which occur when they have risked too much;
- a heuristic model of belief which better simulates the human agents by failing on the types of inferences they find difficult, since it chooses not to expend all of the effort needed to model belief accurately unless it is absolutely necessary; and
- a system architecture which allows agents to choose the next action to take in the dialogue flexibly and according to the Principle of Parsimony, whether that involves planning the dialogue or attempting to recover from a plan failure.

1.3 The Corpus Analysis

Our analysis of a corpus of map task dialogues shows that many of the decisions which agents must make in order to engage in task-oriented dialogue can be characterised as a choice between "risking" completion of some part of the task on the current attempt or expending extra effort in order to ensure success. Agents tend to try to complete the task with as little effort as possible, and so they quite often decide to take risks. Such agents will then need to have strategies for recovering from failures which occur when their risks turn out badly. Analyses of the corpus revealing the nature of decision making in the map task and a number of recovery strategies form the first two major contributions of this thesis.

The HCRC Dialogue Database[2] contains 128 map task dialogues; in this work we examined the first eight dialogues which were collected by the HCRC plus a small set of dialogues from a pilot study used in Shadbolt's work [Sha84]. From time to time in this thesis we will wish to reprint abstracts from the transcriptions. The data from the two different sources are transcribed in slightly different ways; throughout the thesis we preserve the transcription provided by the corpus. The dialogues from the pilot study are marked with a rough measure of the time elapsed between parts of utterances; each plus sign (+) designates one to two seconds of silence. Both sources indicate the interruption of an utterance using angle brackets (< and >), with a slash (/) roughly indicating the place where the beginning of the overlap occurs. Places where the transcription is not clear are indicated with question marks (??).

1.3.1 First Contribution: An Analysis of "Communicative Posture"

Within the map domain there are many different ways that agents can approach the task, leading to dialogues with widely varying structures. For instance, the agents might begin by discussing the objects on the map:

MAP 2

A: right now would you like to describe your picture to me or shall I describe mine or just lead in

B: well I've got start and then next to it pretty much in the right-hand corner and on the same sort of horizontal level there is there are three things an old well deserted village a church

A: aha er where is the old well are they equidistant or

B: well no not really well the old well's nearest to the start point and it's about er let me see south easterly direction from the start point it's about two inches from it

A: and is it on a level with the deserted village

B: er well the top of the well is on a level with the bottom of the deserted village

A: right and then is on mine the church is in the top right-hand corner

B: yeah that's roughly the same as mine not exactly the top right-hand corner but pretty much the top right hand corner

A: below it are some flat rocks

Other agents might decide to simply begin describing the route and deal with difficulties as they arise:

MAP 6

[2]For more information, contact the Human Communication Research Centre, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland. The email address is HCRC@edinburgh.

C: OK, from start point... So what you want to do now is a straight vertical line upwards that will loop round the stone pillars.

K: Right. OK.

C: So the top of your line will be slightly curved.

K: Yes.

C: And it will be about an inch and a half off the top of the page.

We will return to the difference between these two approaches in section 3.2.2.1. In the first case, the route giver is careful to make sure that the agents mutually understand any objects which might come up in subsequent descriptions of the route, whereas in the second case, the route giver relies on his or her partner to interrupt if some utterance occurs in the dialogue which he or she do not understand. Thus the difference between these two approaches is that the first two agents are careful to establish mutual understanding of the objects on the maps and spend a significant amount of effort to do so, whereas the second two agents are careless about mutual understanding and spend more effort after problems occur. This difference in approach is central to the thesis. In chapter 3, we will introduce "communicative posture parameters", based on Shadbolt's work [Sha84], which use this same basic distinction to explain twelve different kinds of decisions which agents make in the map task corpus; the one which we have introduced in these examples is called "difference", since it involves an assumption about how much difference there is between the two maps. The parameters give a theoretical basis for decision making in all parts of the map task, from choosing a description of the route to realising an utterance in natural language. The JAM system uses some of the communicative posture parameters found in the analysis of the human map task corpus in order to allow different computer agents to mimic some of the different human approaches.

1.3.2 Second Contribution: An Analysis of Recovery

The approach taken in the second human dialogue extract above requires the agents to be able to fix problems which arise when they take risks. In these cases, the human agents commonly use what we, following Moore [Moo90], shall term "recovery strategies"; these range from having short subdialogues which convey any information which was missing from an explanation to abandoning an entire description and starting over. The ability to take risks and then recover if things turn out badly is exactly what makes human dialogue so robust, and, in the cases where the risks turn out to be good ones, efficient; thus giving computer dialogue systems the ability to make the same kinds of risks and then recover from failures is an important step towards making their behaviour seem more natural. Chapter 4 identifies a number of recovery strategies and shows them in use in the human corpus. The first two major contributions of

the thesis are analyses of the decisions which the human agents make which involve risk and the strategies which they use to recover from subsequent failures. Together with the Principle of Parsimony, these developments suggest why the human agents make the decisions they do when they are engaged in the map task.

1.4 The JAM System

Given this analysis, the remainder of the thesis shows how the communicative posture parameters and recovery strategies are incorporated into JAM, a working computer program. There are two major contributions which are made in this area: a heuristic model of belief which is more suited to simulating the human agents than previous models because it fails on the same kinds of inferences that humans find difficult, and a system architecture which allows the recovery strategies to be incorporated in a way which is both flexible and easy to modify.

1.4.1 Third Contribution: The Belief Model

Previous methods for representing beliefs in dialogue systems have used logical representations so that the system could represent beliefs nested to arbitrary levels and involving arbitrary numbers of agents. Although this flexibility is desirable where the system designers cannot be sure ahead of what depth of reasoning will be required, it has one major drawback: running the logic of belief after every utterance in the dialogue is slower than it needs to be, since moves in a dialogue introduce predictable changes to the belief system.

If one knows ahead how many agents are involved in the dialogue and how many levels of nesting will be necessary, it is possible to build a fairly simple representation which is easier to maintain. We propose a belief representation which is especially designed to show the regularities of belief changes during dialogue and which can be tailored to different dialogue systems depending on the number of agents and the levels of nesting which will commonly be needed. It proposes assigning to every concept in the domain a state which describes how the agent thinks the other agents stand with respect to the concept. The state designations themselves depend on the needs of the particular dialogue system. For instance, suppose that any one agent can be unaware of P, be aware of P, or be an expert about P. No agents can represent that they are themselves unaware of P, but they can represent that any other agent stands in any of the three conditions with respect to P. Taking a cross product, for two agent dialogues, considering only one level of nesting, there are six possible states: on the first dimension, A can either be aware of P or be an expert about P, and on the second dimension, A

can think either B is unaware of P, B is aware of P, or B is an expert about P. With two agents and two levels of nesting (allowing the representation of facts like "A thinks that B thinks that A is unaware of P"), there are eighteen states. Then, analysing the dialogue in terms of conversational games as in Houghton [Hou86], for instance, there are simple, well-defined transitions in the network of states for each possible move in the dialogue (such as asking a question, giving an answer, or making a statement). For example, asking a question of the partner will, in ordinary conversation, allow one to assume that the partner now thinks that one doesn't know the answer but is aware of the predicate involved. These transitions allow a simple and very efficient way of representing changes in belief that come about as a result of conversational moves. In addition, changes in belief can be propagated within a network of the domain concepts to change the states of related concepts. There are several possible ways of handling uncertainty within the representation. The state transition network can be regarded as a compiled version of a more general belief representation. We also show how the heuristic mechanism can be connected to a more complete belief representation, allowing more complex reasoning when the need arises. The heuristic mechanism allows us to fulfill the Principle of Parsimony concerning the effort involved in belief modelling, since running it requires less effort than running a more complete mechanism but also yields correspondingly less accurate results. Thus choosing to use the heuristic mechanism is another form of risk-taking similar to any of the high risk communicative posture decisions.

1.4.2 Fourth Contribution: The System Architecture

Since we have suggested that an agent's choice of recovery strategy is governed by the Principle of Parsimony, any computer process which is to simulate a human map task participant must be able to choose whichever strategy involves the least amount of estimated effort to complete the task. Very few systems have incorporated recovery strategies at all, and those which have incorporated more than one have chosen which one to use based on a rigid ordering of the strategies. This is the case because such systems have tended to view the recovery strategies as outside procedures which can be called in to "mop up" difficulties when they occur. JAM agents treat using a recovery strategy in exactly the same way as they treat all other possible ways of continuing from a point in the dialogue, giving a uniform mechanism to which the Principle of Parsimony can be applied. In effect, the recovery strategies, along with normal planning and execution, make up a layer of meta-operations which can be planned analogously to domain level plans. JAM agents uses a layered planning architecture modelled on Stefik's MOLGEN [Ste80]. By continuing the abstraction of layers upwards, further levels of operators meta-plan the choice of a recovery strategy and the choice of a way to choose a recovery

strategy. This arrangement has the advantages that it is easy both to change the way in which recovery strategies are employed and to add new recovery strategies, simply by changing the operator definitions at the higher levels. The same style of layered architecture is beneficial more generally in any planning domain where different ways of continuing from the same point in the plan apply.

1.5 The Organisation of the Thesis

The remainder of this thesis is organised as follows:

Chapter two contains a review which describes previous research which has had a substantial impact on this thesis.

Chapter three gives the results of an analysis of the data in the human corpus which yields a detailed set of communicative posture parameters.

Chapter four gives the results of an analysis of the data in the human corpus which yields a set of strategies for recovering from failures in the dialogue.

Chapter five describes all aspects of the JAM system except the two which form major contributions: its model of belief and the layered planning architecture.

Chapter six describes JAM's model of belief.

Chapter seven describes JAM's layered architecture.

Chapter eight gives a number of dialogues generated by the JAM system and discusses them.

Chapter nine gives our conclusions and a number of ideas for further work.

In the next chapter, we turn our attention to a review of the past literature.

Chapter 2

Review of Related Work

Work from a number of fields is relevant to the research described in this thesis. Section 2.1 describes two previous attempts to explain the variety of behaviours possible in human dialogue. It describes Hovy's work on the influence of pragmatic goals on natural language generation, in order to show the variety of choices that are open to communicating agents. It also gives details of Shadbolt's definition of the Principle of Parsimony, about communicative posture, and about his parameters governing the choices which must be made in dialogue, since these are the ideas which underly the central theme of the thesis. Section 2.2 reviews three areas of relevant work on the structure of dialogue: dialogue game theory, a prevalent model of how agents take turns in a dialogue, and a method for simulating both agents in a dialogue. Section 2.3 describes a number of issues that arise in implemented dialogue systems: how to interleave planning and execution and allow agents immediate access to sensory data, ways of classifying the different types of goals an agent might have into independent sets, and methods for controlling meta-planning (or planning about what to plan). Section 2.4 reviews past work on recovery strategies, both analyses and implementations. Finally, section 2.5 reviews work which is relevant to the model of belief presented in chapter 6. This includes work on theorem proving approaches as well as the idea of "degrees of acquaintance" about a concept from the intelligent tutoring systems literature. This review is not intended to be the definitive word on developments in all of these areas, but only to provide the necessary background for an understanding of the work described in the remainder of the thesis.

2.1 Modelling an Agent's Choices

Agents have a great many decisions to make when they engage in dialogue, concerning everything from what goals they have right down to lexical choice. Very little work has been done on explaining why individual agents make the decisions that they do. There are two main pieces of work which have dealt with the issue of such choices: Hovy's PAULINE program and Shadbolt's communicative posture parameters.

2.1.1 Hovy's PAULINE

In his thesis, Hovy [Hov87] begins to account for some of the choices which human agents make while planning a discourse by cataloguing a number of different pragmatic factors which generators need to take into account. The pragmatic information which is relevant includes:

- the interlocutors' factual knowledge (so that a speaker might have the goal of accessing the hearer's knowledge, increasing it, reorganising it, or decreasing it);
- the interlocutors' opinions (so that one might want the topic to seem good or bad to the hearer, whether that is contrary to his or her opinion or bolsters it);
- the interlocutors' emotional states (so that one might have goals to be polite, provoke anger, and so on);
- the interlocutors' goals (so that one might have a goal to activate or deactivate one of the hearer's goals);
- the interpersonal relationship between the interlocutors (so that one might want to affect the hearer's emotions towards the speaker, the "distance" between the interlocutors, and their relative statuses); and
- the conversational atmosphere or setting (i.e., the amount of privacy, background disturbance, and so forth).

These goals are implemented in PAULINE by means of twenty-three different pragmatic parameters with two or three possible settings each. Before a run of PAULINE, these parameters must be set, and the choices which are made affect the resulting text. Of course, this pragmatic information is described at too high a level to be of use to a natural language generator directly. PAULINE incorporates the speaker's pragmatic goals by means of a number of rhetorical goals which directly affect the generator's syntactic decisions. For instance, the rhetorical

goal *formality* has three possible settings (*highfalutin, normal, and colloquial*) and controls the complexity of the syntactic constructions used by the generator and whether or not opportunities for ellipsis are exploited. PAULINE has a set of rules which determines the settings for rhetorical goals based on the pragmatic settings. The relationship between pragmatic and rhetorical goals is quite complex because the pragmatic goals do not have independent effects; typically two or three pragmatic goals working together will determine a rhetorical goal setting. Hovy contrasts a number of texts generated by PAULINE using different pragmatic settings in order to demonstrate that the link which PAULINE uses between pragmatic and rhetorical goals does in fact produce reasonable effects.

Hovy's work bears a family resemblance to the work described in this thesis in that a number of parameter settings control decisions which the planner must make when it is generating text. However, Hovy was unable to give the underlying reason why agents choose to behave in the way that they do or to explore satisfactorily the relationships among his parameters, making them, and the settings used for particular agents, seem arbitrary. This thesis will attempt to explain why agents make some of the choices that they do. We will be able to make this analysis only because whereas Hovy considers a number of pragmatic goals all at the same time, this thesis is only concerned with goals that involve accessing or increasing the agents' information about the map task. Moreover, the decisions which are modelled in this thesis are for the most part unified by involving what sorts of information should be transferred between the agents explicitly and what sorts should be left to be inferred, since these are exactly the aspects of a dialogue which can lead to a failure and the need for recovery. Closely delimiting the area under investigation will allow us to explain better the reasons behind an agent's decisions and to explore the relationships among the communicative posture parameters. However, we share with Hovy the basic idea of setting a number of parameters which govern decision-making in order to generate a variety of behaviours, and any future work which claims to model all aspects of generation may wish to take the other aspects of Hovy's work into account.

2.1.2 Shadbolt's Communicative Postures

Our work is closely based on Shadbolt's analysis of the map task introduced in section 1.1. Shadbolt notes in his thesis [Sha84] that agents use different strategies for task-oriented dialogue in different situations. His strategies primarily concern how much information is transferred explicitly between the two agents and how much is left to be inferred. He calls the different lines along which the dialogues may vary *parameters*. He calls the settings which a particular agent uses for these parameters *communicative postures* and describes them in terms of risk: for the most part, high risk settings leave the agent to infer information and risk the possibility of plan

failure, while low risk settings are more likely to work as planned. He then argues that human agents decide upon their communicative postures according to the Principle of Parsimony, which is "a behavioural principle which instructs processors to do no more processing than is necessary to achieve a goal" (p. 342). Agents choose the parameter settings which they believe will prove most efficient. For each parameter, Shadbolt provides two settings, or *communicative postures*, between which the dialogue participant may choose; a more comprehensive treatment might allow choice along a continuum. The following summary of Shadbolt's communicative postures is taken directly from pages 346-7 of his thesis, with our clarification in square brackets:

High Risk Posture

1.1 Minimal Difference – assume that your partner has the same information about the map features as yourself

1.2 Minimal Specification – in the case of referential instigation, for example, minimally specify the DO [discourse object; in this case, usually a reference to one of the map features]

1.3 Minimal Ontology – select as few objects as possible to relate the task to

1.4 Shared Focus – assume that your partner shares your area of focus

1.5 Minimal Decentering – assume your information is secure; do not take on board incompatible information

1.6 Minimal Feedback – do not expect and do not provide feedback about each other's discourse contributions

Low Risk Posture

2.1 Maximal Difference – do not assume that your partner has the same information about the map features as yourself

2.2 Maximal Specification – in the case of referential instigation, for example, provide as much information as possible so as to identify a new DO and establish its inter-relations with other mutually shared features. As a referential interpreter demand location of new DOs, check the meaning of orientation terms, etc.

2.3 Generous Ontology – be prepared to introduce new DOs into discourse; be prepared to separate out what was previously assumed to be one common feature

2.4 Conservative Focus – move minimally away from the current focus; constantly check current focus of your partner

2.5 Decenter – construct a view of your partner's knowledge state; be prepared for it to be different

2.6 Maximal Feedback – provide constant information so that your partner is reminded of: the task goals, the state of your understanding of his contributions, the information you have on your map etc.

2.7 Hypothesis Formation – Constantly test your partner's representations, and hypothesise about what they might have on their map, what the next step in the task is etc.

Shadbolt's definition of communicative posture provides a clear hypothesis about how agents go about making a great many of the decisions which are necessary in order to complete the map task. Unfortunately, it is not possible using Shadbolt's methodology to show that his categories are either replicable or stable. In chapter 3, we adopt the Principle of Parsimony and much of Shadbolt's terminology, but introduce a revised set of parameters. Our extension to Shadbolt's set is open to the same methodological criticisms; this issue is discussed further in section 3.1.4. The one way in which we improve upon Shadbolt's methodology is by implementing the parameters in a working system as an appeal to computational adequacy. One of the assumptions behind our research is that implementing theories can show up weaknesses or strengths which are not at all obvious from the analysis. Implementation is particularly useful for showing parts of an analysis which are incomplete; for instance, one of the first things which we realised about Shadbolt's analysis when we considered building an implementation is that it did not specify enough information about the possibilities open to the explainee (see section 3.1.1). This caused us to expand his set by, for instance, splitting maximal specification into two parameters. Implementation also made us redefine the parameters so as to make their effects more independent and to place them into sets which correspond to the part of the agent's planning which they affect. The revised set provides a more solid foundation on which to build the implementation found in the JAM system and from which to determine the recovery strategies in chapter 4.

2.2 Dialogue

This thesis relies heavily on previous work on the computational modelling of dialogue. There are three distinct aspects of past work which are important: the notion of "dialogue games", turn-taking strategies, and architectures which allow a program to simulate all of the utterances in a dialogue rather than just those of one agent.

2.2.1 Dialogue Games

This thesis requires a way of abstractly describing the utterances which agents make in a dialogue which notes that dialogues have an internal structure. There are actually many different levels at which one can find "building blocks" in a dialogue; for example, Sinclair and Coulthard [SC75] identify five different size units in classroom discourse, from an entire "lesson", taking up a classroom period, to an "act" which expresses one "idea". However, the most interesting unit from our point of view is what they call an "exchange" and what we, following the dominant terminology, term a "dialogue game." Dialogue games express the fact that, by and large, questions are followed by answers, greetings by greetings, statements by affirmations or denials, and so on. Because all agents know, for instance, what it means when someone asks a question, they can use dialogue games to obtain or share information. There are many different theories of dialogue games which exist, but all of them share the feature of indicating what expectations agents have when they engage in dialogue. The one which we have followed most closely is Houghton's [Hou86] notion of "interaction frames". We have chosen this version because it is intended to be used computationally and because it makes reference to what agents believe as a result of playing a dialogue game, which will become important in chapter 6. It expresses the games in much the same way as plans, in terms of goals, preconditions, effects, and actions. Houghton's frames are described in terms of participants (the agents involved and the proposition expressed), an end goal (the reason for starting the game), an immediate effect (what the agents can be sure they have accomplished when the game is over), preconditions (conditions which must be true before the start of the game), a response (actions which the agents take when the game is initiated), and the type of reply which the initiating agent expects its partner to make. For example, the following is the specification of the MAKE-KNOWN game which Houghton gives on page 90 of his thesis:

Making Something Known

Participants – initiator := <TYPE actor>

addressee := <TYPE actor>

prop := <TYPE fact>

End Goal – know(addressee, prop)

Effect – know(addressee, (know(initiator, prop)))

Precondition – know(initiator, not (know (addressee, prop)))

Response – addressee $\Longrightarrow$ update worldview

initiator $\Longrightarrow$ check acceptance

Reply – accept or reject information

Commentary – The goal of this interaction is to get the addressee to know something. The immediate effect here is that they will at least know that you know it and the precondition is that the initiator believes the addressee does not already know it. The response of the addressee is to attempt to integrate the information into its current beliefs. The initiator waits for uptake of the information which is indicated by the addressee's reply.

Houghton has four games in all; the other three are for asking the addressee for some information, getting the addressee to do something, and getting the addressee's attention. The main difference between our dialogue games and Houghton's interaction frames is that our games sometimes have more than the two moves allowed by his initiation and reply; in some cases it will be useful to allow, for instance, an agent to accept or reject the reply to a wh-question. Allowing more than two moves is quite common (for instance, in Carlson [Car83], Kowtko et al. [KID91], Levin and Moore [LM77], and Sinclair and Coulthard's exchanges [SC75]). Our own dialogue games are described in section 5.4.

Of course, there are many different versions of dialogue games that we could have adapted for use in the JAM system, both simpler and more complex than Houghton's interaction frames and stressing different properties of dialogue. Power [Pow74] implements seven games simply as routines in a programming language; although his implementation is less flexible than Houghton's, it still suffices. Levin and Moore's games [LM77] are completely goal-based like Power's and are described in terms of the process of playing them (recognising what game is being played, pursuing it, and terminating the game). More recently, another goal-based account, Clark and Schaefer's [CS89] "discourse contributions", has stressed the importance of contributions from both agents even during games in which information predominantly originates with one of them. Their work concentrates on the intention of dialogue to establish mutual understanding and allows for different ways of expressing a move to give different strengths of evidence about an agent's understanding and beliefs. All of these versions of dialogue games are oriented towards computational mechanisms, but there are also formulations of dialogue games which are concerned primarily with analysis. Sinclair and Coulthard's exchanges [SC75] are one example. Carlson's [Car83] dialogue game moves have much the same information structure as Houghton's, but are more loosely strung together; rather than giving predictions about what will happen next and thus guiding the addressee's behaviour, his dialogue game rules only specify possible behaviours such as in the rule which he calls "D.answer" — "When a player has put forward a question, an addressee may put forward an answer to it" (page 60). In

addition, Carlson allows moves to be made in the head (by allowing the agent to play dialogue games with a special agent called "Nature") or even unconsciously (by having "Nature" only be aware of the moves, as in the case when one raves deliriously in an otherwise empty room). These properties make Carlson's games admit very diverse dialogue structures but make them difficult to adapt for computational modelling. Finally, very recently Kowtko et al. [KID91] have analysed the structure of dialogues in the map task corpus specifically and created a set of games which the participants use. It would not be difficult to convert their games to a form which is similar to the one which we adopt in section 5.4; the resulting structures would be more detailed than the ones which the system uses. It would be likely to improve the simulation if we were to incorporate their analysis; regrettably, this is left to further work because the JAM system was written before their analysis was complete.

2.2.2 Turn-taking

In section 5.4, we will claim that the model of dialogue employed in the JAM system is a reasonable one because if one makes certain assumptions it fits empirically observed data about turn-taking. Models of turn-taking describe how control of the channel of communication is shifted among a set of agents. The model which we will be adopting for this purpose is that of Sacks, Schegloff, and Jefferson [SSJ78]. In their model, which is designed for ordinary conversation, there are four different *unit-type* (loosely speaking, length) turns which a speaker can make; turns can be sentential, clausal, phrasal, or lexical. They leave open the question of how agents recognise the unit-type in progress in any one move, but provide empirical evidence that agents are able to recognise which type is being used before the turn is complete. Speakers are initially entitled to have one unit, after which a *transition-relevance place* (TRP) occurs. Transfer of speakership can occur at any TRP, and these occur at the end of every unit.

There are two rules which govern transfer of speakership at any TRP and which both allocate the next turn to a particular agent and coordinate the transfer in such a way as to minimise the timing gap or overlap between utterances. These rules are taken directly from [SSJ78] page 13, with clarification added in square brackets.

1. At initial turn-constructional unit's initial TRP:

   (a) If the turn-so-far is so constructed as to involve the use of a "current speaker selects next" technique, then the party so selected has rights, and is obliged, to take next turn to speak, and no others have such rights or obligations, transfer occurring at that place. [One "current speaker selects next" technique involves direct address, as in, "John, what do you think?"]

(b) If the turn-so-far is so constructed as not to involve the use of a "current speaker selects next" technique, self-selection for next speakership may, but need not, be instituted, with first starter acquiring rights to a turn, transfer occurring at that place.

(c) If the turn-so-far is so constructed as not to involve the use of a "current speaker selects next" technique, then current speaker may, but need not, continue, unless another self-selects.

2. If, at initial turn-constructional unit's initial TRP, neither 1(a) nor 1(b) has operated, and, following the provision of 1(c), current speaker has continued, then the Rule-set (a)-(c) reapplies at next TRP, and recursively at next TRP, until transfer is effected.

The dialogue model used in the JAM system fits this turn-taking model, given some assumptions about the flexibility with which we can map JAM dialogue games into surface level utterances. Justification for this claim is deferred to section 5.10.

2.2.3 Simulating Both Agents in a Dialogue

An important aspect of the JAM simulation is that it allows two computer agents to converse without outside intervention. We have chosen this style of system rather than one which converses with a human user for two reasons: it makes the simulation clearer, since particular agents will be consistent in their choices of communicative postures and recovery strategies, and it ensures that the main issues of the thesis are not obscured by the need for natural language understanding. However, our choice does have a complication which is unique to systems that simulate both agents in a dialogue; more traditional systems do not require a way of bringing two agents to "life" on a serial processor at the same time. We have adopted Power's [Pow74] solution to the problem. In his work, which involves dialogues between two computer agents about getting in and out of a room in a world with a door and a bolt, the system starts up by calling a "chair" program which alternately wakes the two agents one at a time. If an agent wants to communicate with the partner, then it passes an utterance to the partner for it to read the next time it is awakened by placing it in the partner's "mailbox". This arrangement is also used in Houghton's [Hou86] work. He considers a "turn" to be one awakening of an agent, but we do not adopt this definition, since, as we shall see in section 5.10, in order to conform to the the Sacks, Schegloff, and Jefferson [SSJ78] model of turn-taking, we must allow turns to occasionally span several awakenings, between which the partner only provides the affirming response or feedback moves. The most significant failing of this arrangement is that it does not allow both agents to "think" at the same time. Nevertheless, Power's approach suffices for

the JAM simulation; our implementation of it is described in section 5.10.

2.3 Issues in Implementing Dialogue Systems

There are three major problems in the implementation of dialogue systems for which we take specific approaches from the literature: the problems of how to combine planning and execution in a way that is reactive enough to account for the human behaviour in the map task corpus, how to characterise the different types of goals which agents have, and how to control meta-actions such as deciding what domain action to take next.

2.3.1 Interleaving Planning and Execution

Most work in planning to date has concerned creating complete plans for some task up to a specific level of detail, without reference to the execution time of the plan. However, the benefits of being able to interleave planning and execution in the face of uncertainty were recognised very early on (e.g., [Sac73]). Young and Simon [YS87] claim that this technique is especially useful in human-computer interaction, because in dialogue all actions are undoable and so they are relatively "safe" to perform even if their consequences are not entirely known; Gilbert [Gil87] claims that such an arrangement is more plausible than the traditional model as a description of human planning in general. Ambros-Ingerson's IPEM [AI86] has the most thorough treatment of interleaving planning and execution to date. It operates on a partial plan that may have certain properties known as *flaws* by applying plan transformations called *fixes*. One possible fix is executing an action, and so execution is given the same status as any other planning activity. IPEM uses a scheduler to decide which fix to apply next, providing for a set of different problem solving strategies including both the traditional method of planning and then executing and interleaving planning and execution. The architecture of the JAM system, as described in chapter 7, fits IPEM's structure. In particular, we have a level (the *strategy* level) of operators which encode all of the possible next planning actions which an agent can take, and can be thought of as fixes; scheduling is performed by a further level (the *mode* level). For the most part, our strategy level operators do not correspond directly to fixes which are discussed by Ambros-Ingerson, since they are specifically tailored to simulate the human corpus. There are also many different fixes and scheduling techniques used in IPEM which do not have corresponding strategy level operators in JAM since they are not useful for demonstrating our central thesis.

JAM also differs structurally from IPEM in that JAM has the same two-phase operation

separating perception and action that can be found in a number of AI systems. Each time an agent is awakened using a Power-like [Pow74] communication model as described in section 2.2.3, before it begins to decide what actions to take using a mechanism like IPEM's, it processes all of the communications from the partner which have accumulated since the last time it was awakened. This process is described in section 5.11. These communications can be considered to be the JAM agent's sensory data. It is quite common for sensory data and reactive actions for "emergencies" to be treated in a separate component from the main planner; such an arrangement may be more psychologically plausible than integrating the two (a claim made by, among others, Bartle [Bar86]). Bartle gives three different ways in which perception and action can be kept separate in a working system and dangerous flaws in a plan (such as an upcoming division by zero) can be avoided. Kaelbling's architecture for intelligent reactive systems [Kae87] divides perception and action into two different components and continually processes sensory data even during long computations. McCalla, Reid, and Schneider [MRS82] describe a taxi-driver simulation, ELMER, which uses situated action rules in the form of daemons on sensory data in order to avoid obstacles such as red lights and pedestrians. Not as much control of perception is necessary in the JAM system, since the top level control structure ensures that new sensory data is present only when an agent is first awakened and never arrives after the agent has begun planning. However, JAM could easily be adapted to use an approach similar to any of these ways of handling perception if a top level control structure were contrived which required it.

2.3.2 Characterising an Agent's Goals

Most researchers agree that even an agent's direct goals to achieve a domain task can be divided into different categories. We will use such a division in chapter 3 in order to separate our communicative posture parameters into sets which affect different parts of a JAM agent's planning process. The most usual distinction which is drawn is between *strategy* ("what to say") and *tactics* ("how to say it"), in Thompson's terms [Tho77]. In working systems, there are a number of different sets of terms which describe similar distinctions. Ferrari and Reilly [FR86] divide dialogue into "substantive real life" goals (*S-goals*) and "communicative" goals (*C-goals*) which can be used to bring about completion of the S-goals. Cawsey's [Caw89b] *content* and *presentation* plans for the EDGE system and Litman and Allen's [LA87] *domain* and *discourse* plans for plan recognition divide in roughly the same way as S-goals and C-goals. Iida et al. [IYA90] introduce a "three typed pragmatics" for recognising plans in spoken dialogue which divides Litman and Allen's discourse plans into three separate parts: *dialogue plans* (plans to engage in cooperative dialogues), *communication plans* (corresponding to one utterance

goal), and *interaction plans* (characterised by turn-taking and playing the role of "discourse expectations" or dialogue games). Their divisions closely resemble the analysis of classroom instruction dialogues done by Sinclair and Coulthard [SC75]. Appelt [App85] further divides what is roughly Iida's communication plan into four different hierarchically arranged levels: *illocutionary goals* (such as informing or requesting), *surface level goals* (such as asserting or commanding), *concept activation* (which involves the planning of descriptions which will refer felicitously to the necessary concepts), and *utterance creation* (involving, among other things, lexical choice). Since Appelt's levels have been used in a working system and are more specific than the other divisions, we will be adopting them for use in our own work. Divisions similar to Appelt's will be introduced in chapter 3 in order to demonstrate that the communicative posture parameters described there form independent sets affecting different levels of planning.

2.3.3 Meta-Planning

There are two basic ways in which past researchers have controlled decisions about what planning action to take next. In the first, the planning agent uses a blackboard-style approach ([HRHR79]), either specifically representing meta-goals and meta-actions (as in Wilensky [Wil83] and Macmillan and Sleeman [MS87]) or allowing a supervisor to control the meta-planning, in effect, by having it choose among different planning options (as in Sabah [Sab90]). Another option is to explicitly cycle through a sequence of meta-actions, either choosing the first applicable one that is found (as in the TEAMWORK project [Dor85]) or choosing from a set of applicable actions based on some measure of priority (e.g., Georgeff and Ingrand [GI89] and Ramshaw [Ram89]). The strategy which will be adopted for the JAM system is a variation on the cycling approach which allows greater flexibility by having agents also choose different ways of cycling through the possibilities in different situations. The way in which this is implemented is described in chapter 7. The JAM system's architecture was designed by analogy to MOLGEN [Ste80], a planner which uses a layered problem solver in order to move through a search space of possible molecular genetics experiments. Although MOLGEN uses its layers to solve a different kind of problem than JAM, the way in which the two systems break down their problems into smaller pieces are very similar. We will first discuss why Stefik designed MOLGEN with a layered control structure, and then describe his choices for each of the layers.

2.3.3.1 Why MOLGEN has Layered Control

Stefik's motivation for providing layered control for his planner stems from the fact that quite often, scheduling the tasks involved in problem solving is difficult in its own right. Layered

control is a flexible and "potentially steerable" approach to scheduling. Most problem solvers are agenda-based; that is, they work on a fetch execute cycle in which the objects retrieved, and the retrieval and selection procedures which make up the interpreter, can be quite complicated. The problem is knowing how the interpreter should decide which objects to select. The decision can be arbitrary (e.g., selecting tasks in the order in which they were created), but in general better behaviour is produced if selection is based on some domain-dependent notion of "priority" or importance. Stefik's solution, based on work by Hayes [Hay75], is to apply the same problem solving techniques to the scheduling problem as to the domain problem, forming a meta-layer for the planner. If scheduling the tasks for this problem solver is difficult, then yet another layer can be added, and so on, until the top level interpreter is trivial. Choosing a task to schedule at any level of the planner is either an easy problem or it is broken down using the higher levels of the planner. Additionally, the interpreter for any level, which consists of all of the higher levels of the planner, is structured in such a way that it is easy to change the way in which tasks are scheduled and to examine the way in which the interpreter works.

2.3.3.2 MOLGEN's Layers

One of MOLGEN's novel features is that the top level control of the planner is a simple "select and execute" loop; all planning strategies are incorporated into the planner by using the layers, of which there are four. The lower layers plan the task and the upper layers meta-plan the planning. The bottom level, the domain space, contains normal laboratory operators coding actions that molecular geneticists can take in an experiment, and abstractions of those actions. These are the normal planning operators that would occur in a non-layered architecture. The second layer, the design space, contains operators that control hierarchical planning and other techniques that would ordinarily be part of the planner's algorithm. For instance, there are operators **propose-goal** and **propose-operator** which choose laboratory goals and operators to add to the plan, **refine-object** which which satisfies plan constraints, and **refine-operator**, which develops a plan to a lower level of abstraction. The third layer, the strategy space, contains operators that allow the planner to use either a least commitment or a heuristic approach to planning. Operators at this level **focus** on building a particular part of the plan, **resume** work on a previous part of the plan, **guess** at solutions to constraints, and **undo** plans if they become overconstrained. Finally, the interpreter (the top layer of the planner) simply chooses which of the options at the strategy level to take. If the interpreter always focuses and resumes work rather than guessing, it takes a least commitment approach. If it guesses at solutions, then it uses a heuristic approach and takes the chance of having to undo some work. This layered approach to planning allows different planning strategies to be tried with minimal

changes and elegant control.

When MOLGEN is run, it is invoked at the top level, the interpreter, with the goal of constructing an experiment that satisfies a given set of constraints. The interpreter uses a least commitment approach to planning as far as possible, and then if the plan is still underconstrained, uses a heuristic approach. In the least commitment approach, the interpreter applies the **focus** operator, which tries to design new steps and carry them out. It does this by asking each of the design operators (such as refining an operator or propagating a constraint) what work they can do given the current state of the plan and preferring those which propose new goals and operators over those which do not. If some design task is underconstrained, then it is suspended for later consideration. When the **focus** operator runs out of things to do, then the interpreter has the **resume** operator go through the different suspended tasks looking for one that can be restarted. If only the **focus** and **resume** operators are used, then the planner does not commit itself to any decisions which could cause the plan to become overconstrained. However, if the interpreter can not complete the plan using only these two operators, it must adopt the heuristic approach to planning, using the **guess** operator to make a plausible decision which will allow the plan to be developed further. If at any time guessing causes the plan to become overconstrained, then the **undo** operator is used to backtrack.

MOLGEN uses its layers to explicitly break up the problem of planning in a fairly complex formalism into decisions such as what kind of global planning strategy to use (least commitment or heuristic), which activity to use to extend the plan (e.g., add a new goal or apply an operator to a goal), and which domain level operator to use. JAM uses its layers to break up a problem as well, but instead of solving the problem of how to plan, it solves the problem of how to decide what to do next in the dialogue. Thus its layers are different from MOLGEN's; they decide which way of choosing a way to continue the dialogue to use (such as "communicate as early as possible", or "if possible, don't replan"), which way to continue the dialogue (e.g., planning, communicating, or attempting some kind of recovery), and which domain level operator (or dialogue game) to use. In fact, all of the decisions which are made in MOLGEN belong properly to the one way of continuing the dialogue in JAM which is encoded in the **plan** operator. However, JAM's planning is sufficiently simpler than MOLGEN's so that it does not need to break down planning into different levels. The relationship between MOLGEN and JAM is one of analogy. We adopt from Stefik his way of looking at problem solving and apply it to a problem with different characteristics. The layered architecture used in MOLGEN is appropriate for JAM for the same reasons that it is appropriate for MOLGEN: the layers provide flexible control for the system by breaking up the problem of choosing the next move to make in the dialogue into smaller problems using meta-planning. JAM's layers are described in

section 7.4. A worked example of a JAM run which demonstrates the layered control is given in section 7.5.

2.4 Recovery Strategies

In chapters 4 and 7, we introduce a number of strategies which are used by agents in the human corpus to recover from plan failures, and describe an agent architecture which is especially designed to allow flexible access to these recovery strategies. There are three distinct issues from the past literature which affect our treatment of the recovery strategies. The first concerns classifications of different kinds of failure. The second is definitions of recovery strategies which directly influence the analysis of recovery strategies used in the corpus, as found in chapter 4, and their JAM implementations, as described in section 7.2. The third issue is how past researchers have incorporated recovery strategies into working systems, so that we can compare their techniques with the flexible approach used in the JAM system, as introduced in section 7.4.

2.4.1 Classifying Misunderstandings

A number of researchers have tried to categorise misunderstandings. Their work is useful in categorising the different kinds of conditions under which an agent might require a recovery strategy. Most work has been done on classifying and recovery from referential failures (e.g., Goodman [Goo86]; McAllister [McA87]); work at this level of detail is beyond the scope of this thesis. More generally, Schuster and Finin [SF83] give five different conditions which can lead to misunderstandings in a dialogue. The conditions are expressed from the system's point of view in an interactive system:

1. The system and the user may use different terminology.
2. The user may fail to make an intended inference, make an inference other than the intended one, or make an invalid inference.
3. The user may misunderstand a referring expression.
4. The user may not see a step as an alternate procedure rather than as an additional step to an existing procedure.
5. The user may fail to recognise the issues because an explanation is too general or too specific.

These classifications suffer somewhat from being a bit vague. For instance, one might consider conditions three and four to be special cases of condition two where the inferences involved are in some sense not in the "real world" but in the realm of discourse. The recovery strategies which we will be describing in chapter 4 do involve failures which fall into Schuster and Finin's classes. Condition one may arise due to a high risk setting for a parameter which we shall identify in section 3.2.1.2 as *partner modelling*. Conditions three and four are caused by high risk settings for parameters which we shall identify in section 3.2.3 as *specification* and *description resolution* in the former case and *context articulation* and *context resolution* in the latter; the most usual recovery from them falls generally in the category which we will define in section 4.2.3 as *repair*. Condition five may arise as the result of high risk settings for the *ontology* and *ontological resolution* parameters defined in section 3.2.1; recovery from it can be achieved using the strategies *elaboration* and *omission*, which are defined in sections 4.1.3 and 4.1.4, respectively.

2.4.2 Past Approaches to Recovery

We will be adopting a number of recovery strategies almost directly from the literature. We will be using Moore's [Moo90] definitions for *repair*, *replanning*, and *reinstantiation*:

Repair: A plan may be repaired after it has been executed and failed if at least one of the prerequisites of the plan is found not to be satisfied. In this case, the repair can be brought about by fulfilling the failed prerequisites and executing the plan's operations again.

Replanning: If some part of a plan has failed and there is some other possible plan which will achieve the same goal (i.e., one which uses a different plan operator), then the failed plan may be abandoned in favour of the other plan. Replanning can occur not only at the lowest level goal which has failed, but also at any of the higher level goals for which that low level goal is being performed. For instance, if a plan to get an agent to know the location of the swamp fails, and that goal is being performed in order to get the agent to know how to complete a particular part of the route, then the agent may wish to substitute a new plan for that part of the route which does not involve knowing the location of the swamp.

Reinstantiation: If a plan has failed due to a problem with the value of one of the plan's variables and there are other possible values for that variable, then the plan may be reinstantiated with a different value for that variable and then reexecuted. For example, if a plan for nailing a bookcase together fails because the nail is bent in the process, one

way to recover is to begin again with a different nail. For reasons which we shall give in section 7.2.3, this strategy does not occur in the map task but is implemented as one way of replanning in the JAM system.

Moore's recovery strategies are defined simply but specifically, and repair and replanning match behaviours found in the map task corpus quite well. However, in more complex planning domains, where plans cannot be changed as flexibly as in dialogue, more special purpose versions of the recovery strategies may be necessary (e.g., involving cleaning up the adverse effects of a failed plan before executing a replan). Although we will be adopting Moore's definitions of the recovery strategies, we will not be adopting her method of incorporating them into a system, for reasons which we will give in the next section. There are also recovery strategies which will be introduced in chapter 4 which Moore has not discussed. We will require a simple version of plan recognition in order to implement one of these, *goal adoption*, which we will define in section 4.1.1 and our implementation of which we will describe in section 7.2.5. There are several different approaches to plan recognition available ([KA86] and [Car90] provide good reviews); we have chosen to adapt part of Allen's approach only because, given the rest of the JAM system, it seemed the simplest to implement. Allen's plan recogniser uses two different techniques. The first, which we do not adopt, starts from some expected goals and searches for a plan for one of the goals which includes the observed action. The second applies the plan construction rules which the partner might have used in reverse to the observed action, in effect, putting the system "in the partner's shoes" in order to determine why the partner might have taken the action. This part of the plan recogniser uses inference rules such as "If S believes A has a goal of executing action ACT, and ACT has an effect E, then S may believe that A has a goal of achieving E." (pg. 113), which is very similar to the rule which the JAM system uses. However, the JAM system's method for recognising plans is much simpler than Allen's, and provides less coverage, since the point of including it is to create a better simulation of the human behaviour in the corpus, not to improve upon work in plan recognition in its own right.

2.4.3 Using Recovery in a Program

Even very early work in planning suggests the use of various different recovery strategies, usually called "replanning". STRIPS and PLANEX are combined in a system for controlling the SRI robot [FIIN72] which can reuse arbitrary subsequences of operations from the original plan when replanning after execution failures. This is primarily an efficiency measure, however, and is not often applicable. Hayes [Hay75] improves upon this approach by recording in the structure of the plan details about why each of the planning choices was made; when a separate

execution monitor signals specific failures, then the replanning can invalidate some of the decisions and all parts of the plan relying on them. Then replanning involves filling in the gaps in the resulting partial plan. Ambros-Ingerson's IPEM [AI86] has the capacity to replan more generally, but only when it is strictly necessary to do so and only on the lowest level failed goal. Peachey and McCalla [PM86] build plans which allow several routes through lessons in an intelligent tutoring system, and so the system can replan in a limited way when its initial plan fails. In addition, if none of the possible routes through the lesson work, the system can take the improved student model (which results from having the failed dialogues) back to the planner and generate a new plan for the lesson. All of these systems implement restricted plan recovery but do not need to choose among different strategies because each has only one way of handling plan failure.

Wilkins' SIPE [Wil88] implements recovery (which he terms "replanning") more comprehensively by using a *problem recogniser* to diagnose particular failures when unexpected situations occur, and then applying one of eight different *replanning actions* to effect the recovery. The replanning actions work by manipulating the plan in such a way that it contains new unsolved goals; then the plan can be sent back through the planner to be specified well enough for reexecution. These actions combine to form reinstantiation, replanning, and repair as defined by Moore [Moo90], and another strategy, *repetition*, which is defined in section 4.2.1. They can also take advantage of serendipitous effects and allow some other forms of recovery which work with failures which are too complex to be expressed in JAM's planning language (which is described in section 5.6). However, Wilkins' replanning actions can not be used to reconstruct all of the recovery strategies found in our analysis of the human corpus.

None of the systems which can use more than one type of recovery have very flexible methods for deciding which one to use, nor has there been much analytic work on the choice of strategies in human dialogues. In Moore's work [Moo90], there is a strict ordering placed on the possible forms of recovery which decides which of them to apply. The system attempts repair first, reinstantiations next, and, if none of these work, begins all of the different possible replanning options working from the most specific (bottom level) failed goal upwards. SIPE [Wil88] similarly has strict orderings on the application of the replanning actions, although most of them only apply to particular kinds of plan failure, so there are not usually many possible actions to take. McTear [McT85] suggests from empirical evidence on simple surface level repairs in conversation that people tend to favour self-initiated recovery over relying on the partner to initiate the recovery, as long as no recovery has yet been initiated. We argue in section 7.1 that a strict ordering on the recovery strategies cannot possibly cover the range of behaviours found in the human dialogue corpus, and suggest that by applying the Principle of

Parsimony we can choose the strategy which will allow the entire task to be completed with the lowest estimated cost. This seems a likely way to explain McTear's observation (since self-initiated recovery can reasonably be expected to have a lower cost than some possible future recovery initiated by the partner) while providing an underlying principle that can explain other aspects of human choices as well.

2.5 Belief Modelling

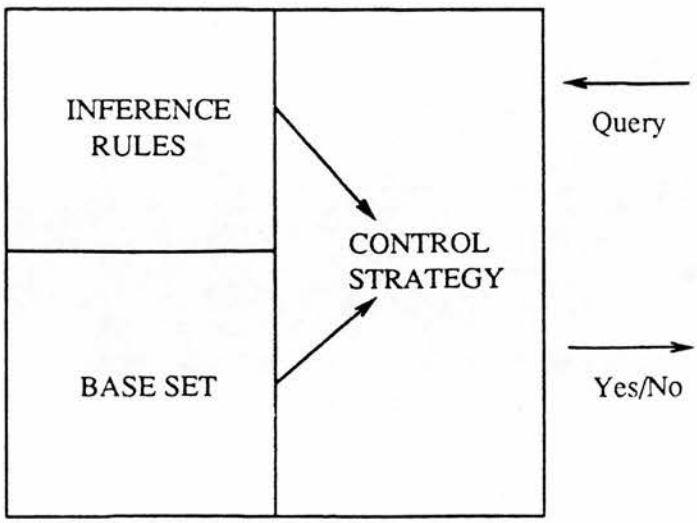
The vast majority of research in belief modelling has been based on standard theorem proving techniques. These methods are formal, elegant, and relatively easy to build and manipulate. However, there is no natural way within them to limit reasoning about beliefs to that which humans can do. In chapter 6, we present a belief model which takes into account the intuitive organisation of dialogue and domain knowledge in order to restrict belief updating in a way that better simulates the behaviour of the human agents. In this section we review the theorem proving approaches against which we will be arguing and provide some background information which is relevant to the new belief model.

2.5.1 Theorem Proving Approaches

At an abstract level, Konolige [Kon86] gives the best description of how a theorem-proving based belief model can be incorporated into a computer agent. He defines a *deduction structure* to have the form given in figure 2.1. It has a knowledge base of beliefs, a set of inference rules, and a control strategy for applying those inference rules. When a query is put to the system, the system tries to match the query to some fact in the knowledge base or to derive the query from some sequence of inference rule applications. Then the *belief set* of the system is that set of sentences for which the system answers "yes". Konolige assumes that the system has a fixed, short amount of time in which to answer each query, allowing the system to be logically inconsistent. Konolige also allows nested beliefs to be represented by, for instance, agent A believing that agent B has a deduction structure of a certain form. In this case, B is not required to use the same inference rules or even language to express beliefs as agent A. Because Konolige's way of representing belief as deduction structures is general enough to encompass all of the possible theorem proving approaches, it is useful as a way of comparing the different options. However, his concern is to explain the cognitive structure of AI planning programs and not to model people, so he does not specifically try to simulate human belief updating.

There are many different approaches to belief modelling which fall into the category of theorem

Figure 2.1: Konolige's Deduction Structure



proving and which can be described using Konolige's framework. Despite the fact that they rely on a process which has no basis in psychology whatsoever, most of the variations have been invented in order to better match the kinds of inferences that people make. Israel [Isr85] investigates a number of standard modal logics in order to determine how specific axioms relate to likely inferences for humans to make if the operators of necessitation and possibility are read to mean knowledge and belief. Different axioms add properties such as introspection. Experimentation with them might allow the creation of a system which is baroque but well suited to modelling people. Both autoepistemic logic [Moo84] and default theories [Rei90] have been used to describe the kinds of inferential jumps which people make; more recently, Konolige [Kon87] has shown that each can be embedded in the other and so the choice between them is not crucial. Konolige also [Kon88] introduces a default logic with a hierarchy of evidential spaces so that intuitions about some inferences blocking others can be encoded; this formalism has been applied to speech acts [AK88a] and reference [AK88b]. Levesque's logic of implicit and explicit belief [Lev84], Frisch's inference without chaining [Fri87], and the work of Fagin and Halpern (for example, [FH88]) specifically tackle the problem of creating a representation in which agents are not necessarily logically omniscient. Finally, Ramsay [Ram87] and Ballim [Bal87] look at problems which arise in nested belief spaces and involve representing that other agents know information which the self does not. These techniques have some success in capturing the kinds of inferences which come naturally to humans; their major limitation is that the closer their behaviour gets to that of humans, the more convoluted their logics become, until they are so complex that they are very difficult to work with and not at all plausible. We believe that this is a necessary characteristic of the "logic in the head" approaches. In chapter

6 we will argue for an alternative approach for which we make no psychological claims but which takes into account the organisation of an agent's knowledge in order to limit inferencing in a way that better simulates the human agents.

2.5.2 Degrees of Acquaintance

Our own belief representation, presented in chapter 6, is very different from the theorem proving approaches which we have just described, but provides a heuristic solution to the same problems. In section 6.1, we will be introducing the term "degrees of acquaintance" to refer to all of the different kinds of understanding (coarsely speaking) that an agent can have with respect to a domain concept. For instance, we might want to distinguish between agents that are unaware of a concept, aware of it but do not know much about it, have enough knowledge to execute plans using the concept (a position which Steel and Reichgelt [SR90] formalise in a hybrid of epistemic and dynamic logic and term *vividness*), know everything there is to know about a concept, and so on. We will have more to say about the positions which an agent might want to represent in section 6.1. The notion of degrees of acquaintance is related to that of stereotypes used to start off a model of an agent in an overlay model for an intelligent tutoring system; for instance, systems quite commonly designate some agents as novices and some as having more knowledge than that, and then associate a set of concepts to each stereotype which designates those concepts which the agent is likely to know (e.g., [Cla87], [Sel88]). However, in the usual usage, each agent is either assumed to know or be ignorant of a particular concept; we extend this concept so that agents can have different degrees of acquaintance with each concept. In addition to the idea of different degrees of acquaintance, we also borrow from user modelling the idea of heuristic rules which can be used to reason about another agent's knowledge, such as the following, which plays a role in the JAM system:

> If the agent believes that P is vivid for another agent, then the agent believes that there is some way of dividing P in which all of the subparts of P are vivid for that agent.

Kass [Kas88] discusses rules which are very similar to this one and some of the others that we introduce in section 6.7.

JAM's belief representation works by associating standard belief updates with each of the possible dialogue moves using a finite state transition network. Very recent work at the University of Rochester [AS91] suggests a somewhat less general version of the same type of process for keeping track of plans in a cooperative dialogue system; it attaches state designations to plan

fragments which have arisen during the dialogue. These states represent whether the plan fragments have been adopted by one or both of the agents, allowing the system to modify a plan's state based on moves in the dialogue. The Rochester work is just beginning, but we suspect that there will be some overlap with the ideas presented in chapter 6.

2.6 Summary

The novel work of this thesis is presented in chapters three to seven. Roughly speaking, the following past research is relevant to each of the chapters:

For chapter three (Communicative Posture) – Shadbolt's work on communicative posture parameters and Hovy's work on the effect of pragmatic goals on natural language generation in PAULINE, as well as ideas about how to classify the different types of goals that an agent might have;

For chapter four (Recovery Strategies) – Past work on recovery strategies, especially that of Moore, and Allen's techniques for plan recognition;

For chapter five (The JAM System) – Dialogue game theory (especially Houghton's interaction frames), Power's work on simulating both agents in a dialogue, the Sacks, Schegloff, and Jefferson model of turn-taking, and ways of allowing an agent immediate access to sensory data;

For chapter six (The Belief Model) – Theorem proving approaches to modelling belief plus some ideas from intelligent tutoring systems and user modelling about heuristic belief updating, and Steel and Reichgelt's definition of "vividness";

For chapter seven (The Layered Architecture) – Previous approaches to planning how to plan (especially Stefik's MOLGEN) and to interleaving planning and execution.

The novel work of this thesis can be seen most clearly in relation to work left undone by Shadbolt, Moore, and Stefik. We improve upon Shadbolt's analysis of communicative posture in the map task corpus by completing a more extensive analysis and by looking at the use of recovery strategies in the human corpus. In addition, we implement a number of communicative posture parameters and recovery strategies in a working system, JAM, which demonstrates their viability. We improve upon Moore's way of incorporating recovery strategies into a working system by developing an architecture which allows agents to choose their next actions based on the amount of effort which they believe remains to complete the task, given each of the

possible next actions. This provides a principled choice and more flexible behaviour, as well as making the architecture easy to modify. At the same time, we demonstrate Stefik's very general approach to problem solving in a markedly different domain from molecular genetics that requires the ability to interleave planning and execution. The remainder of the thesis is divided into an analysis of the human map task corpus (chapters 3 and 4), an explanation of how the JAM system works (chapters 5, 6, and 7), example output from the JAM system (chapter 8), and conclusions and further work (chapter 9). Next, we turn our attention to the role of communicative posture in the human dialogues.

Chapter 3

Communicative Posture

This chapter examines the way in which communicative posture affects the behaviour of agents in the human map task dialogues. Shadbolt's communicative posture parameters and the Principle of Parsimony have been reviewed in section 2.1.2. Section 3.1 extends Shadbolt's analysis of the communicative posture parameters by taking into account the decisions of the explainee and by dividing the parameters into different sets for different discourse planning stages, making their operation more independent and therefore simpler to control in a working system. Section 3.2 lists the parameters used in the extended analysis. This analysis provides a theoretical basis for the implementation of the parameters and lays important groundwork for both the analysis of recovery strategies, found in chapter 4, and their implementation, discussed in chapter 7. Within our analysis of communicative posture, examples of high and low risk settings from the human corpus are given for each parameter. Only a selection of the communicative posture parameters given are implemented in the JAM system. This chapter records which parameters are implemented, but discussion of how they are implemented is deferred to section 5.12. Finally, although all of the examples which we give in section 3.2 are from the map task dialogues, in section 3.3 we argue that the analysis applies to interactive, collaborative problem solving more generally.

3.1 Extending Shadbolt's Analysis

Shadbolt's analysis of communicative posture, as reviewed in section 2.1.2, covered important groundwork for this thesis by providing a small, corpus-based set of parameters and invoking the Principle of Parsimony as an explanation for why map task participants choose the parameter settings which make them behave as they do. We will be adopting both Shadbolt's method of

observing the parameters and his understanding of the Principle of Parsimony, and for the most part we will adhere to his analysis of the parameters. However, there are two basic ways in which Shadbolt's parameters can be improved. The first is by taking into account that agents don't just give explanations, but also hear them, and the second is by dividing the parameters into subsets, each of which affects a different part of the agent's process for planning what to say in the dialogue. In addition, in preparation for the analysis which forms the rest of the chapter we clarify what we mean by "success" and "risk".

3.1.1 Parameters for the Explainee

All of Shadbolt's parameters focus on the route giver, since the route giver takes a more active role than the route follower in most of the dialogues in the human corpus.[1] All of them, except possibly decentering, have to do with building an explanation. Decentering at least partially has to do with recovering from an error in an explanation once the partner has heard it and indicated that something was wrong with it. However, dialogue takes two agents; as well as having explainers, it also involves explainees. Explainees need parameters modelling things like how carefully they disambiguate referring expressions, focus movement, and changes in context. These parameters are usually more important for the route follower, since in this task most explanations are about the route and use information that only the route giver has. However, the route follower often produces explanations of where objects are, especially for objects that do not appear on the route giver's map. In cases where the route giver is uncooperative or too highly cognitively loaded to plan all the descriptions, sometimes the route follower produces hypothetical descriptions of routes or the locations of objects to test out on the route giver. For instance, consider the following extract:

MAP 9
A: just draw what I said + right you go over the big river + right
B: right over the big river right + there's not a bridge on it where's the bridge ?
A: dinnae ken
B: is it up from oxbow lake?

This happens more often than one might expect; sometimes the best way for an explainee to disambiguate a description is to enumerate all the possibilities and ask the explainer to choose one. In these cases, the route follower uses the parameters for constructing explanations and route giver uses the parameters for understanding them.

[1] Strictly speaking, this isn't true, since part of maximal specification is "as a referential interpreter demand location of new DOs [discourse objects], check the meaning or orientation terms etc.". However, in practice variations in the dialogue which can be introduced by the explainee are ignored.

3.1.2 Grouping the Parameters by Planning Stage

Shadbolt's analysis of parameters, besides missing out variations introduced by the explainee, fails to justify what parameters are on the list or relate the parameters to each other. It seems reasonable that the set of parameters ought to divide further into subsets which affect different stages of an agent's planning. For instance, some parameters ought to affect task planning (i.e., in this domain choosing a description for the current section of the route or for the location of an object), others to affect creating a discourse plan which expresses the task plan (and by this we mean what Thompson [Tho77] calls "strategic generation"), others to affect realising that discourse plan (or "tactical generation"), and still others to affect what decisions are made when a plan goes wrong. A parameter set that divided into these sets would involve less interaction among the parameters and would have the practical advantage that planning could be done in independent stages. It is clear in which category some of Shadbolt's parameters belong. His ontology parameter clearly affects only task planning, since it has to do with picking out objects relevant to the current explanation. However, some of his parameters cross over these planning stages. His difference parameter, for instance, affects both how an agent chooses a task plan and how the agent organises the discourse to explain that plan. A more independently structured set of parameters would introduce no disadvantages to Shadbolt's analysis, but would be a more reasonable computational model of what occurs in the human corpus and would fit in better with traditional views of natural language generation.

3.1.3 Caveats about "Success" and "Risk"

Before we proceed, we should explain what we mean by both "success" and "risk". For some tasks, it is absolutely clear whether or not a given solution completes the task successfully. For instance, if the task is to get a message across town by noon, then the message either reaches its destination by noon or it doesn't. However, for many tasks there is a tradeoff between the amount of effort expended on the task and the quality of the result. For instance, if one is writing a paper, then (up to a point) the more effort one spends on it the better it will become. Shadbolt [Sha84] describes this kind of task as having "open texture" success, meaning that the task has no objective measure of success. The map task is one such task; the agents are told to "describe it [the route] exactly because it's the only safe route", but not how exact they must be. A variant of the instructions which would give the agents an objective measure of success might tell the agents to reproduce the path within a particular precision (say, for instance, one centimeter on the piece of paper). In open texture tasks, agents must weigh the amount of effort needed for particular solutions against the quality of the result and choose the

solution with the best value for effort. This produces very interesting differences across human agents in sections of the map task where there is a great increase in effort required for a small improvement in the quality of the solution. Unfortunately, it also causes complications for the analysis of communicative posture, since it is difficult to gauge what forms a "low" or "high" risk position without knowing what amount of success the agent is aiming for. Exactly the same behaviour may appear as the result of a "high" risk position for an agent with exacting requirements for success as for a "low" risk position for an agent who is not so exacting. We can only describe a behaviour as being "low" or "high" risk with respect to a particular criterion for success. In the analysis of the human dialogues, we ignore this problem and judge all of the behaviours as if the agents had the same criterion for success. This complication does not arise for the computer dialogues because all of the agents have exactly the same idea of what it means to complete the task.

We also need to clarify what we mean by "risk". Using a high risk strategy does not mean that the agent is risking not being able to complete the task, since agents always have the opportunity to attempt recovery from any parts of their plans which have failed. In terms of completing the task, it is no more risky to take a high risk strategy and expect to do more recovery than to take a low risk strategy in the expectation that less recovery will be required. What is risked in high risk strategies is successful completion of the task *on the current attempt.* Risk in this sense is related to the amount of preparation which the agent makes for the current attempt; low risk agents use more effort initially, while high risk agents risk having to use extra effort to recover from failures due to less stringent planning. It is often more parsimonious for agents to use high risk strategies than to use low risk ones, because with a high risk strategy the agent might "get away with" the initial behaviour; otherwise, agents would never use the high risk strategies. Of course, using low risk strategies does not guarantee that recovery will not be necessary, anyway; they just tend to make recovery necessary less often. Despite the fact that the term "risk" is confusing, we shall continue to use it because it does convey the basic nature of the parameter settings; low risk settings are designed to work without further effort whenever possible, while high risk settings risk failure of the initial plan.

3.1.4 Methodological Issues

In section 2.1.2, we mentioned that the methodology which we use in analysing communicative posture is susceptible to criticism on (at least) two grounds: we have not shown that other analysts can use the parameter distinctions which we describe, making it unclear whether or not the categories are replicable, and we have not shown that our set is complete or "psychologically real" in any way, making it unclear whether or not the analysis is stable. These are tricky

problems, not just for our analysis but for all analyses of its type, and we will not address them properly. It would be possible to at least make some claim of replicability by adopting a technique from Kowtko et al. [KID91]; we could give descriptions of our parameters to a number of people who are not familiar with them and see if they analyse the same dialogue in the same way. Although this would undoubtedly bring up some interesting points about the parameters, we have not had time to do this work. Stability is an even harder question, since it is not clear how one could show that there are no other decisions which agents make during task-oriented dialogue on the basis of effort and risk or that the parameters could not be divided up in another way. We do address both the issues of replicability and stability indirectly by appealing to the computational adequacy of the analysis. The implementation supports the hypothesis that the parameters are replicable because it clarifies what the parameters do and how they affect the agents' behaviours. The implementation supports the hypothesis that the analysis is stable in two ways: first, it shows that the parameters actually "work" in some sense because the implemented versions simulate the associated human behaviours, and second, it forces us to think about all the different decisions which are necessary to a dialogue agent, highlighting parameters which might otherwise not have been noticed in the corpus. Nevertheless, our failure to deal with these methodological issues satisfactorily mean that we can not make as strong claims about the relevance of our analysis as would otherwise be possible.

3.2 The Parameters

The current set of parameters is constructed to extend Shadbolt's set in such a way that it will provide a sound theoretical basis for a computer simulation of the map task data. We have identified twelve parameters, which are grouped into four sets; the remaining sections describe each of the sets, giving the parameters and examples of the different settings from the human corpus. Figure 3.1 gives a summary of which parameters fall into each of the classifications, to which of Shadbolt's parameters they are most closely related, and whether or not they are implemented in the JAM system. Some of the implementations of the parameters in the JAM system only simulate the most important effects of the parameters and not all of the behaviours described in the analysis. More details about how the parameters are implemented and how they affect the JAM system are given in section 5.12.

Figure 3.1: The Communicative Posture Parameters

Our Parameter	Shadbolt's Closest Parameter	Implemented in JAM?
Task Planning Parameters		
Ontology	Ontology	yes
Partner Modelling	Decentering	yes
Ontological Resolution	Ontology	no
Discourse Planning Parameters		
Difference	Difference, Ontology	yes
Coherence	Focus	no
Utterance Realisation Parameters		
Context Articulation	Feedback	yes
Context Resolution	Feedback	no
Focus Articulation	Feedback	no
Focus Resolution	Feedback	no
Specification	Specification	no
Description Resolution	Specification	no
Meta-Planning Parameter		
Plan Commitment	Decentering	yes

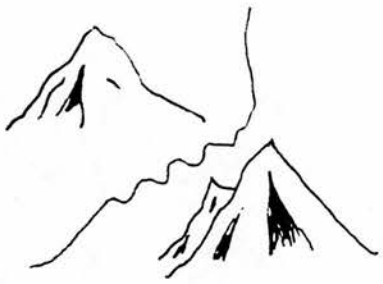
3.2.1 Task Planning Parameters

The task planning parameters (ontology, partner modelling, and ontological resolution) determine the particular task plan which is chosen by the agents. In the map task, they affect the descriptions of the route and of objects on the map which the agents use. Since these parameters affect the way in which the task is planned, they must be implemented in a way that is domain dependent. However, the parameters themselves are not domain dependent; they apply in any task-oriented dialogue domain which divides into concepts which one or both of the participants understand. For instance, in the domain of describing electronic circuit diagrams, the concepts might be the different parts of a circuit and ways of explaining a circuit's function like the "water flowing through a pipe" view or the description using Ohm's law. Whatever the domain, the relevant concepts are all of the different parts of the domain plus the ways in which they can be explained to the partner.

3.2.1.1 Ontology

Ontology controls the concepts to which an agent refers when she explains part of the task or the task domain to the partner. The Principle of Parsimony, as reviewed in section 2.1.2, says that in tasks in which differing amounts of effort give differing amounts of success, agents may use heuristic approaches to completing the task rather than expend a great deal of effort in order to achieve the optimal solution. The ontology parameter reflects the agent's choice of

Figure 3.2: The Path Through the Mountains in Map 9



the amount of effort. The higher the agent's standards for the task, the lower risk her ontology setting will be. In high risk posture, agents only refer to concepts which are easy to describe, taking the risk that important details will be lost. In the map task domain, this involves using simple descriptions which do not refer to very many objects, as in the following extract for the map given in figure 3.2:

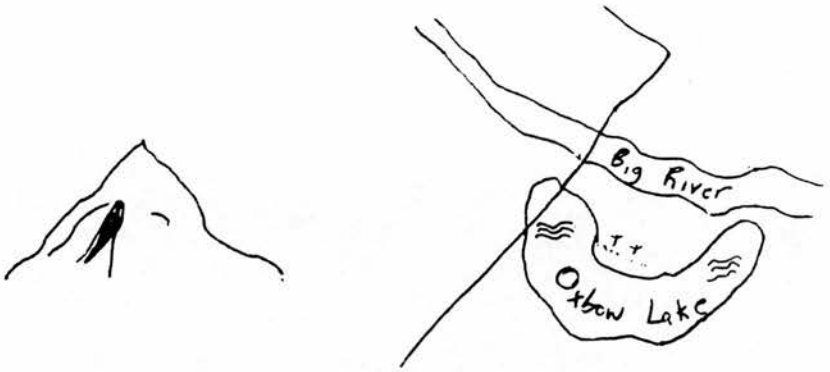
MAP 9
N: yeah ++ and then go + south west and round the waterfall
V: round the bottom of the waterfall + OK
N: North east until you get to the bridge + to the river
V: that's the big river half way there
N: yeah
V: straight north east
N: um yeah
V: yeah OK
N: you cross the river

This is a fairly minimal approach; agent N accepts the consequences of this dialogue as satisfactory, and, of course, agent V, not having any further information, draws the route as given in figure 3.3.

In low risk posture, on the other hand, agents use whatever concepts are necessary to describe the route accurately, even if these concepts are quite complex, as in the following extract:

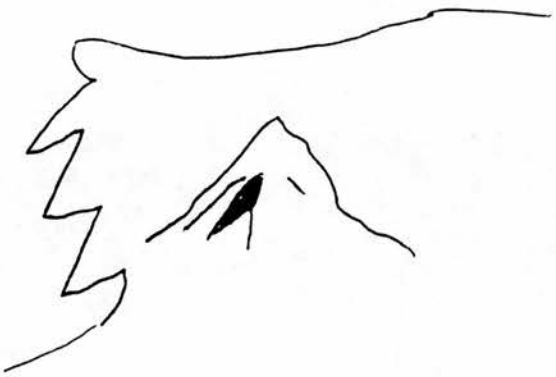
MAP 9
G: so + as you're coming up towards + the base of the mountain that YOU have +
S: yeah

Figure 3.3: V's Map



G: you have to go + in a s- in a squiggle like that (laughs (—) squiggle) erm +
S: so + am I + on the west side or the east side I'm on the west side of the mountains presumably? + the bottom
G: you're + just below the mountain actually
S: right I'm just going
G: approximately right in the middle but what you're trying to do is head towards + the west
S: right
G: you're trying to go past the mountain+ and ++ it's necessary to zigzag + erm ++ four times + if I can put it that way + erm
S: now what + which way do I veer now
G: right + so so you come to a point then + you suddenly zig down + and zag up + if I can put it that way (hah!)
S: OK + so it's
G: four times + right? ++ so that you're going from + about approximately the middle of the diagram up + north + west + to to the + just beyond the bottom of the mountains +
S: OK so + I (—) do I veer + to the
G: You're veering off to your right + northwest
S: yeah + I know + the erm + the zigzag
G: Yeah + the zigzag is going in that direction + yeah
S: But which way does it start?
G: Oh
S: The edge (of the) west + versus the east
G: Oh I see which way you mean + sorry + it veers + actually southwest + to start with
S: yes
G: the first + do you understand?
S: uhum
G: Right + so it's + down up + down up + down up + down + and its going up + the last one you should be just below + you know + below the mountains + coming out beyond the mountains + right? + do you have + er a bridge crossing Big River

Figure 3.4: S's Map



This monumental effort yields the results given in figure 3.4.

According to the Principle of Parsimony, the ontology setting which is chosen by an agent will partly depend on how the agent determines value for effort. An agent will choose low risk ontology if she believes that the amount of effort involved is "worthwhile" because it will lead to a sufficiently better completion of the task. In the dialogue extracts just given, it is possible that G had decided that it was necessary to keep closer to the path than N thought necessary, for whatever reason. It is also possible that N realised that to explain that section of the route accurately within the precision attempted by agent G would mean resorting to inches and angles, which would be a formidable undertaking. In any case, the agents weigh value against effort in choosing their ontology settings. This is made clear in dialogues where the participants discover that a description will take more effort than they originally thought. For instance, in the following dialogue the agents abandon a task plan for one of lower quality:

MAP 10

L: ah right + erm + oh yes + er + I have a crashed plane marked here + can I + check this + my crashed plane is ABOVE + it's in the BASE of the quadrant + top right hand imaginary quadrant of the + erm + picture + yes er + that SOUNDS too high for me +

K: er

L: + because my cacti + my cacti + are about + oh + I would say + about two inches at the moment or one and a half inches below the + imaginary horizontal + er you know the half way line or the vertical dimension of this + er + piece of paper

K: mm + where we fouled up + I think was + when we + calculated from the mountain

L: OK + mm + yes + so I should shift up my cacti a little

K: OK

L: does does this MATTER really or can we just sort of
K: I- I doubt if they're worried
L: hha!
K: yes
L: OK we'll just .....? so + we both have a crashed plane and it's just above the imaginary horizontal
K: OK + yeah + if you divide the whole page + the whole map + horizontally in half
L: mhm hm
K: your crashed plan is about + an inch above +
L: yeah
K: er the base of your cactuses is about an inch below that
L: yeah + OK + wa- we're near enough

This example shows that as the agents realise how much effort is involved in a particular task plan, they may decide to abandon it for an easier (or harder) one by making the ontology simpler (or more complex). In fact, as we shall see in section 4.1.4, this behaviour is an example of a particular strategy for recovering from failures in the dialogue which we term "omission". The ontology parameter is implemented in the JAM system using a simple measure for determining the "complexity" of each of the possible task plans, as described in section 5.12.1. Then high risk agents prefer less complex task plans than low risk agents.

3.2.1.2 Partner Modelling

A *Partner Modelling* parameter controls whether or not the agent uses a model of the partner when constructing descriptions. This parameter does not control whether or not the agent builds a model of the partner in the first place; that is controlled by the difference parameter. We did not find examples in the corpus of agents deliberately referring to concepts which they know their partners do not share. In effect, this entails that low risk difference strategies tend to make very high risk partner modelling strategies unprofitable. Since the bulk of the effort goes into modelling the partner, it would be unreasonable to build the model and then not use it. Therefore, the important factor in recognising whether a strategy is low or high risk is how the agent uses concepts which may or may not be known to the partner. In low risk postures, the agent rates more highly those descriptions which refer to concepts thought to be mutually known than those for which no such information is available. For example, consider the following extract:

MAP 10
G: have you got a graveyard
R: yes over to the
G: over to the left about three inches from the volcano
R: yes I have on the same line +
G: yes and just left up the graveyard have you a ruined village

R: no I haven't
G: well OK never mind ++
[... intervening dialogue with no mention of the ruined village]
R: yeah OK and then you are going to turn due west basically for about one inch again heading towards the graveyard
G: ok about one inch
R: for about one to one and a half inches now what you are going to do is make a big curve
G: mmh hhm
R: right round the graveyard keeping about one quarter inch +++ away from the graveyard as you do a big curve and you end up coming out of the graveyard around the other + side of the graveyard heading south east ++ is that clear

Here low risk difference makes G ask whether or not R understands references to the objects, and low risk partner modelling makes G construct a description of the route through that section which does not refer to the missing ruined village. At least, it seems reasonable that G is putting special effort into constructing such a description; without reference to a partner model, a more usual description of that section of the route is given in the following extract:

MAP 10
V: head south for one and a half inches just at the side of the volcano then ++ two inches to the east - to the west sorry ++ then curve up round the graveyard if you have got it - up round the north side of the graveyard and down between the graveyard and the village ++ yeah

This, then, is the corresponding high risk partner modelling strategy, where the agent is not concerned about a model of the partner, thereby using less effort, at least in the short term.

In the JAM system, the partner modelling parameter is implemented simply with reference to an agent's belief about its partner's beliefs. Agents with low risk settings for partner modelling prefer descriptions which, all other things being equal, only refer to objects which they are sure the partner has, while high risk agents do not distinguish between objects which they know the partner has and those about which they are uncertain whether or not the partner has them. More details about the JAM implementation of this parameter can be found in section 5.12.1.

3.2.1.3 Ontological Resolution

Ontological Resolution determines whether or not explainees ask explainers about concepts that are closely related to the concept that is being explained. It usually improves the explanation if the explainee either asks the explainer if related concepts are relevant, or suggests hypothetical ways in which they might be relevant and asks the explainer to accept or reject them. However, this takes more effort than simply accepting the explainer's explanation as it is. In this domain, since the basic concepts are things like objects and routes, concepts can be related either by

semantics (i.e., having the same label or semantically related labels) or by location. There are three versions of low risk ontological resolution for this domain. The first, and most general, is mentioning any objects which look crucial to some description that the partner has given but weren't included in the description. Agent M uses this version in the following extract when hearing a route description:

MAP 6
H: The start is just to the right of the antelope, and first go straight up due north, if this is facing north.
M: Mhm.
H: And then, curves round to go under what I've got marked as desert, which presumably is your sand dunes.
M: <So,/
H: It goes
M: do I go round the stone pillars?>
H: Yes.

In this extract agent M uses spatial reasoning to come up with a hypothetical description of the route and check the description with agent H.

Another low risk strategy, which requires the agent only to have knowledge about when concepts are somehow related to each other, has the agent ask how related concepts fit into the explanation, as in the following example when K mentions the hills:

MAP 5
C: Now what you want to do is loop round the crocodiles crossing the river as you go round.
K: Right, then do you mean loop north of them or loop south of them?
C: Er, north, about half an inch away from them.
K: OK, if I do that, as I cross the river I come into hills.

K then leaves C to deduce how the hills fit into the description of the route.

Another version of low risk ontological resolution, possible only in domains like the map task where the number of concepts is not only finite but small, is used in just one of the human dialogues. It involves mentioning every concept which the partner has not mentioned to make sure that none of the remaining ones are important, as in the following extract:

MAP 1
[after finishing the route]
B: right because I don't have that and what about other things on your map any woods old temples hills do they come into this route
A: hills woods pyramid no old temple
B: so I should be there and the old well and the plane crash is the only thing that we had to navigate round that we weren't sure about
A: yeah
B: ok

This in some ways is the opposite of the very low risk difference strategy, where the agents establish all possible concepts *before* starting the route description. Low risk difference and ontological resolution are the only ways in which A can get a model of B's map; A must either explicitly build a model of B's map, or B must offer information about the differences between the two maps. Although there are three versions of low risk ontological resolution, there is only one version for high risk. High risk agents ignore objects on their own maps which are not mentioned by the partner. Thus low risk ontological resolution takes more effort, at least in the short run, than the high risk setting. The ontological resolution parameter is not implemented in the JAM system. In order to be used effectively, it requires at least one of the agents to be able to handle spatial reasoning, and that is beyond the scope of this research.

3.2.2 Discourse Planning Parameters

Discourse planning parameters control the way that discourse is organised once an agent already knows what task plan she will use. There are two discourse planning parameters: difference and coherence. The task plan only expresses very vaguely what should transpire in the discourse by specifying what the core of the explanation should be. The difference parameter fills in details about optional utterances which could be added to that core explanation which might cut down the total amount of effort spent on the explanation. After the difference parameter has specified what the different parts of the discourse should be, the coherence parameter affects their ordering in the resulting text.

3.2.2.1 Difference

Difference controls the amount of difference that participants assume there is between their own model of the domain and the partner's model. If an explanation will contain the first reference in the dialogue to some concept in the domain[2], it can sometimes save effort to make sure that the partner understands that concept *before* beginning the explanation. For instance, consider the following beginning segments of dialogues:

MAP 9; High Risk Difference
N: Start in the south east corner and go north west past the palm beach and around the swamp

MAP 9; Low Risk Difference

[2] Although in the map domain the difference parameter affects only first references to objects, in more complex domains that involved longer dialogues, it might affect subsequent references if the agent forgot whether or not the partner knows about the object. Forgetting by either agent complicates the partner model considerably and is beyond the scope of this research.

M: OK
B: yes
M: erm if you look to the right of or up from the start
B: mmh
M: erm there's a palm beach do you have that?

In the first extract, N assumes that the partner will have both the palm beach and the swamp. He uses definite descriptions to refer to these objects, and he does not ask the partner first whether or not the objects appear on the map. In the second extract, M makes no such assumptions, and so must ask B whether or not they share the palm beach. M also uses an indefinite referring expression, "*a* palm beach", to indicate that she does not assume that B will understand the expression as a unique reference, or, indeed, as a reference at all. In most of the dialogues, where agents adopt low risk difference settings, they also adopt indefinite referring expressions for objects until their partners have indicated that they understand the references. However, it is also possible to use definite referring expressions for the same use, as in the following extract:

MAP 10; Low Risk Difference with Definite Referring Expressions
K: right you have the start marked
L: yes mmm
K: alright d'you have the volcano marked?
L: yes

For the purposes of choosing a difference setting, the type of referring expression is not important; it's just that some difference settings rule out either definite or indefinite expressions. Agents with a high risk difference strategy must use definite referring expressions, since they assume that their partners will understand a particular reference. Agents with a low risk difference strategy may use either definite or indefinite referring expressions, as long as on their own maps and in the current focus the referring expressions denote unique objects. Difference only determines whether or not the agent assumes the partner knows about an object for each object on the map. However, our discussion of the parameter has left open the question of where in the dialogue the agent asks about the objects. In the preceding examples, agents chose possible descriptions of the route segment first and then built models of their partners for the appropriate objects. If agents think that there will be many differences on the maps, they may decide instead to build a model of the partner which contains all the objects they can foresee they might use *before* choosing route descriptions. In the last extract, this was in fact what agent K was doing; K and L discussed every object on both maps except for the cave and then closed that part of the dialogue with

MAP 10

K: I think + that we can + start now
L: fine

Since the cave is the only object nowhere near the route, the cave is the only object which is not discussed. In runs with other agents, variations on the same technique are used where the agents establish every object on the maps or every object on one of the maps. Any of these techniques involves a very low risk difference setting, since the agents assume that their choices of route descriptions will be significantly improved by having detailed models of their partners.

Such low risk difference strategies are not currently implemented in the JAM system. A simple version of the difference parameter is implemented which makes low risk agents ask their partners whether or not they have any objects in a description they intend to use if the objects have not been discussed before. Then if the partner gives a negative response, the agent can clarify the reference or abandon the current discourse plan, depending on which requires less future effort. The JAM system also only allows the difference parameter to affect first references for map objects rather than for concepts generally. For instance, in the map dialogues agents might need to discuss the meanings of concepts such as "make a horizontal line" or "go around the route", but such discussions are not allowed in the simulation. Details about the JAM implementation of the difference parameter can be found in section 5.12.2.

3.2.2.2 Coherence

The *Coherence* parameter controls how carefully the participant orders explanations. The low risk setting uses some method for deciding how to order ideas once they have been considered essential to the explanation. The high risk setting is based on just saying things in whatever order the agent thinks of them, whether that is the best organisation for the utterances or not. In domains where agents have quite a bit to say before they expect feedback from the partner, or where putting utterances together is tricky, a low risk setting can be expected to improve the resulting text substantially. Current work in text generation (for instance, schemas [McK85], Rhetorical Structure Theory [MT87], and focus trees [MC88]) addresses ways of building such low risk strategies. However, in the map task domain it seems likely that the order in which agents think of things to say is usually the same as the best order in which to say them, and so expending extra effort on ordering the information does not generally lead to better performance. Route giving agents in the map task seem to think of the route as if they were following it from start to finish, and then organise their explanations chronologically. We found no dialogues in the corpus where the route giver described segments of the route in anything but chronological order. There were cases where, in the description of a single segment, the route

giver described a specific position on the map and then asked the route follower to connect that position with the end of the route so far, in effect describing the single segment backwards. For example, consider the following extract for the volcano map:

MAP 10
K: + now + we have (a) difficult part + erm +++ find + the mountains in the bottom right hand part
L: yes
K: of the sheet +++ find + the leftmost point of the mountains that are defined??
L: yes mhm
K: OK at about one half inch to the left of that point
L: hmh
K: put a dot
L: mhm
K: and + connect that dot with + where you ended up to the right of the waterhole +
L: er + right ++ have done

It seems likely that K used this technique because the segment's shape was simple, but its end point was hard to find. Using a nonchronological description in this domain appears to be highly marked, and forced agent K into explicitly articulating the current context by using the phrases "find" and "connect that dot".

The coherence parameter seems important in text generation as a whole, but does not play a large role in the map task. Therefore it is not implemented in the JAM system.

3.2.3 Utterance Realisation Parameters

This set of parameters controls planning of the discourse at a level after choosing a task plan and organising the discourse have already been done. They control realisation of the utterances; in the human corpus, that includes things like word choice, but in the JAM system, where the agents use an artificial language and can't misunderstand concepts like "around" or "between", the choices to be made are much simpler. They have to do with synchronising focus and context between the two agents. For each parameter affecting the explainer, there is one affecting the explainee. This happens because these parameters have to do with tactical generation, and for each aspect of tactical generation, there must also be some aspect of understanding by which the explainee decodes the message. For the purposes of this research, we will define context as the current set of intentions (that which answers the question "what are the agents doing?") and focus as the set of concepts which the agents use in the discussion. The parameters handle synchronising changes in context and focus where they occur.

3.2.3.1 Context

In the map task, there are two basic contexts: the agents can be working on drawing a segment of the route or locating some object. Additionally, there are a few higher level contexts, like working on establishing every object on the map or working on drawing the entire route. These higher level goals are always composed of a series of the two kinds of lower level goals. Therefore changes in context are fairly simple to track, and the only mistakes come in thinking one is supposed to be drawing instead of looking or vice versa. There are two parameters related to context changes: *context articulation* for the explainer, and *context resolution* for the explainee. They handle making sure context changes are unambiguous and disambiguating context changes, respectively.

3.2.3.2 Context Articulation

There are two ways in which explainers can be "careful" to articulate context changes. The first is to include meta-comments, or utterances which involve not the task itself but how the task is to be approached, in the dialogue whenever they are not sure the partner will be able to figure out what the current context is, as in the following extract:

MAP 6
M: Right. You have got to
H: Yes
M: direct me. Erm, from looking at the maps before... On my map, just to see if there are any substantial differences,
H: Mhm
M: or how similar they are, there's a river...
H: Yeah

Here M is very careful to tell H exactly what his current intentions are. It is harder to give high risk examples. Agents often fail to declare their intentions, but it is not possible to prove that they did not consider doing so and decide that it was unnecessary. However, the following extract, repeated from section 3.2.2.2, includes one of the most difficult contexts available in the domain, so the explainer involved can be considered high risk for not articulating it:

MAP 10
K: + now + we have (a) difficult part + erm +++ find + the mountains in the bottom right hand part
L: yes
K: of the sheet +++ find + the leftmost point of the mountains that are defined
L: yes mhm
K: OK at about one half inch to the left of that point
L: hmh

K: put a dot
L: mhm
K: and + connect that dot with + where you ended up to the right of the waterhole +
L: er + right ++ have done

The context in this case is drawing a segment of the route backwards, and, despite L's affirmation, he did not understand that he should draw anything on the paper. One supposes that K expected L to be able to follow the context change because K warned L that "now we have a difficult part" and used the word "connect" for the first time. This brings us to the second technique for articulating context changes. In the map domain, context meta-comments tend to be found only with higher level context changes. More basic context changes are signaled by adopting different language for each of the different contexts. Agents who use this technique reserve specific words for use when they have specific intentions, in effect redefining the words more narrowly for the purposes of the dialogue. The words "draw" and "go" usually signal the route drawing context, while the words "look" and "find" signal the intention of locating an object. Once explainers have chosen words and had them understood by their explainees, they tend to use the same ones again.[3] In more complex domains (e.g., in math texts), explainers explicitly make such definitions for intentions that occur frequently, and use meta-comments for less frequent intentions. Although it may seem that such word choices come naturally to explainees, and therefore don't count as low risk strategies because they involve little or no extra effort, there are examples in the corpus of dialogues where the route follower's choice of action verbs seems to confuse the route follower about the current context:

MAP 9
R: Ok you are going to come about one inch below that palm beach
G: um um
R: + and ++ you are not quite horizontal you are taking a slight curve up towards um the swamp ++ not obviously going into it
G: well sorry I have not got a swamp
R: you have not got a swamp?
G: no
R: OK
G: start again from the palm beach
R: from the palm beach alright, have you got trees drawn on your palm beach
G: yes
R: alright
G: so I am to leave the palm beach eh
R: I am going to draw the swamp for you first
G: ok
R: ok um um if you go about two inches across from the top of the tallest tree in the palm beach
G: so I go to the palm beach

[3] Simon Garrod has some ideas about why this is the case; see section 9.2 for details.

R: no don't draw the route just now I am just showing you where the swamp is sorry

R uses words inconsistently; for instance, R uses the word "draw" first for one basic context and then the other. R also uses several words for the same context, forcing G to decode the context using semantics each time rather than simply matching the word with the same context indicated in the past. Examples like this one indicate that there really is extra effort involved in making good word choices which some agents are either unable or unwilling to expend.

Since the JAM system uses an artificial language that only includes one word for each of the basic contexts, word choice strategies for context articulation are not implemented. The simulated agents can only choose whether or not to explicitly tell their partners what their current intentions are using meta-comments. Details about this limited implementation can be found in section 5.12.3.

3.2.3.3 Context Resolution

Context Resolution is a parameter used by explainees to determine the partner's current context if it is ambiguous. As for all disambiguation, there are four basic options available: the explainee can guess a context and take no further action, the explainee can guess a context and indicate what she thinks the current context is to the explainer, the explainee can indicate to the explainer that there is a problem (either generally or specifically concerning context), or the explainee can list all the possible contexts and ask the explainer to choose one. The third and fourth options are never used in the corpus for context resolution, although they are used to resolve other ambiguities. Since the first option leaves no evidence for outside observers, it is impossible to tell whether explainees have used it or whether they have simply failed to notice the ambiguity. Evidence for the second option is found in the corpus, although explainees tend to indicate their context choice indirectly, as in one part of the high risk context articulation extract, reprinted here:

MAP 9
R: ok um um if you go about two inches across from the top of the tallest tree in the palm beach
G: so I go to the palm beach

There are no examples in the corpus of explainees beginning a discussion of their context choices, for instance, by asking the partner "do you mean that I should draw?" It seems likely that because context choices in this domain are so simple, agents feel enough confidence about their choices that indirect references to them are sufficient.

No context resolution strategies are implemented in the JAM system because the artificial language leaves no facilities for agents to utter statements which leave the context in doubt.

3.2.3.4 Focus

Focus changes in the map task are more complicated than context changes. There are two common lines along which focus can be expected to change during the course of the dialogue. Agents may use some idea of what is in focus spatially to refer to objects near the current section of the route or near objects under discussion. This is the usual way of introducing new objects, since the map task naturally involves spatial clusters of objects. Changes of focus along these lines tend not to require any special signaling on the part of the explainer, since the first place an explainee looks for an object to match a referring expression, if such an object is not in focus, is the spatially neighbouring objects to the objects in focus. Objects also may potentially be brought into focus by being semantically related to objects that are in the current focus. For instance, if a "wood" is in current focus, then all other "woods" are likely to be brought into focus, and if the "swamp" is in current focus, then "crocodiles" may be brought into focus naturally. This kind of focus movement is useful because these objects are exactly those which are likely to be misunderstood in the current discourse, and so mentioning the semantically related objects can be a low risk strategy for making sure that the agents agree on the current focus. In fact, Shadbolt ([Sha84], pg. 355) claims that the route follower is more likely to change focus to include semantically related objects than the route giver, because the route follower often has no idea where the route is going and therefore has fewer spatial clues. Agents sometimes signal focus changes of this type, since they introduce ambiguity and therefore are inherently tricky. Low risk agents always signal focus shifts which are unexpected, such as long distance movement on the map to centre on an object which is not semantically related to an object in the current focus, since such shifts are potentially confusing for the explainee.

Shadbolt includes as one of his parameters one that handles focusing; in high risk, agents move focus however they please and do not explain their focus shifts to their partners, and in low risk, agents change focus as little as possible and check their current focus against the partner often. The current work uses two parameters to handle focus whereas Shadbolt uses only one. The part of his parameter which makes low risk agents move focus minimally is part of our coherence parameter in the discourse planning stage, since it involves how the different things which must be said are ordered. McCoy and Cheng's [MC88] focus rules could be used to implement this, although the JAM system currently does not implement this parameter. The other part of Shadbolt's parameter, providing clues to the partner about focus shifts which

are hard to follow and continually checking that the partner has the right focus, belongs to the utterance realisation stage of the planner. This part of his parameter, which is also not implemented, may involve adding some meta-comments to the discourse, but it mostly involves changing the way that references to particular objects are realised.

3.2.3.5 Focus Articulation

Once agents have decided to change their focus in a particular way, communicative posture determines whether they assume that the partner will be able to follow the focus change or whether they do something to indicate the shift to the partner. The high risk strategy for focus articulation says to assume that the partner will be able to follow, and so high risk agents do not include any explicit focus information. In the low risk strategy, agents provide explicit focus information for their partners both for all focus shifts which might be hard to follow and also at some other times, simply as a check. Sometimes the focus shifts are signaled by meta-comments in the same way that context changes are, but more often focus shifts are signaled by adding some kind of modifier to the referring expression for the object involved in the shift which specifies how the focus should be changed. Here we consider the focus information to be a meta-comment if the information applies to a series of objects, as in the following example:

MAP 6
M: And above the river there I have five landmarks, erm, a picket fence on the left...

We consider other examples, which include only one object, to modify individual referring expressions, even if they may serve the purpose of changing the partner's focus permanently. However, the distinction fades if one allows compound objects (e.g., "the area above the river") into the ontology, and so we will not differentiate between the two ways of realising low risk focus articulation in the remaining examples.

Simple focus shifts to include spatially nearby objects are almost never explicitly signaled; since this kind of focus shift is the most prevalent, agents can expect their partners to follow them. Focus shifts which involve large jumps on the layout of the map are sometimes signaled, as in the following example, where H uses low risk focus articulation:

MAP 8
H: Then, a few inches below, directly below the caverns I have a cactus, sticking out in the middle of nowhere. Then, going back up to where the lighthouse is again, erm, in the opposite corner to that, if you like, the left ... top left hand corner is a ruined arch

This is an example of where the explainer, H, uses the area of the map to explain the focus shift. Another technique is using movement along the x or y axis:

MAP 10
K: The best relation I can describe it is that the finish point is level with what I have as a dead tree in the other corner of my map.

However they choose to explain the new focus set, low risk agents indicate to their partners that an unusual shift is taking place. On the other hand, high risk agents choose not to indicate large jumps in focus. In the following example, the stone pillars have been discussed by these agents earlier in the same dialogue, although in an entirely different part of the discourse:

MAP 6
M: Sand dunes in the middle, s-... and stone pillars on the right
H: right
[...approximately eleven turns of dialogue]
M: And then near the bottom of the page, er, an oxbow lake.
H: Yeah. Does that have a picket fence above it?
M: No it doesn't.
H: No it doesn't. Right OK, er, do you ... did you say you had stone pillars
M: No stone pillars.
H: somewhere? No stone pillars.
M: Oh, erm, I have stone pillars in the top of the map, the top right hand side.
H: Right.

Here H's intention is to use the stone pillars as an anchor to get back to the top of the map, since she has objects in that region which they have not yet discussed. Here, her high risk focus articulation strategy leads to M having to repair the dialogue by changing his focus and (through low risk focus resolution, as it will be defined in section 3.2.3.6) telling her of the change he has made.

The focus articulation parameter is not implemented in the JAM system. A simple version of it could be installed by enriching the map representation with some notion of map regions and x and y coordinates, and using them to articulate difficult shifts in the low risk agents.

3.2.3.6 Focus Resolution

Focus Resolution, like the other explainee parameters which occur in pairs with parameters for the explainer, allows the explainee to ask the explainer if there is a focus shift which has not been signaled but which the explainee does not understand. There are no examples in the corpus of low risk agents directly asking about the current focus with questions like "Are you in the middle of the map?" or "What part of the map do you mean?" Instead, explainees

tend to tell their partners what they are using as the current focus, so that their partners can correct them if necessary. We've already seen one example of this at the end of the "stone pillars" example in section 3.2.3.5. In other cases, sometimes the explainee mentions a possible focus change:

MAP 10
K: erm + do you have the desert above the dead tree
L: Not + my my + ??? + my desert's in the top half of the picture ++ stony desert

In any case, the purpose of a low risk focus resolution strategy is to force the explainer to disclose the current focus. Agents with high risk focus resolution strategies choose the most likely option when they find a focus shift is ambiguous.

The focus resolution parameter is not implemented in the JAM system.

3.2.3.7 Specification

Once the agent assumes that the partner shares a particular focus set, the agent must construct a referring expression for the object. *Specification* controls the number of properties given to disambiguate new concepts (e.g., objects in this domain). Agents using high risk specification strategies use whatever expressions they think of first to refer to concepts; low risk strategies involve spending more effort to build referring expressions suited to the current partner. The best referring expression depends on the objects which are known to the partner. A low risk referring expression should both uniquely pick out the concept from the concepts in focus both uniquely and minimally; that is, it should not contain any components that are unnecessary in uniquely denoting the object. The low risk strategy requires uniqueness so that the hearer of the expression won't have to disambiguate the expression. It requires some kind of minimality because, as Reiter shows [Rei90], overly specific referring expressions can confuse the hearer by causing false implicatures. However, the problem of finding a unique expression which is truly minimal is NP-hard; A more relaxed definition of minimality, for the purposes of implementing a low risk strategy, could be defined with respect to a mechanism like Dale's [Dal88] "discriminatory power" with a heuristic ordering on the kinds of properties to determine which ones are added to the description first, which is the approach taken by Reiter and Dale [RDng]. The specification parameter is not as important in this domain as it is in other domains, since agents can refer to most objects using their printed labels. That means that there isn't the same element of word choice in this domain as in many others; objects really only have one major description.[4] The labels usually denote the objects uniquely and minimally within the

[4]There are cases where agents deliberately choose not to use the labels, as in one agent's calling the palm beach "wee palm trees", as they appear in the map symbol. These variations are beyond the scope of this

current focus set unless the agents choose to talk about objects that are semantically related. This parameter is not implemented in the JAM system.

3.2.3.8 Description Resolution

Description Resolution, working opposite *specification*, controls the way that agents find the referents of descriptions. In low risk posture, explainees might ask for clarification of ambiguous descriptions. In medium risk posture, explainees might use an algorithm like Goodman's [Goo86] to determine the most likely referents. In a high risk posture, explainees might use the first object they find fulfilling the description. This parameter is not implemented in the JAM system.

3.2.4 Meta-Planning Parameter: Plan Commitment

So far the set of parameters has included controls for making decisions in the task planning, discourse planning, and utterance realisation stages of the dialogue. However, there is a stage of planning more abstract than any of these which controls decisions about how planning should continue. This stage controls decisions about whether it is best to continue with the same plan or abandon the current plan for a new one. There is only one parameter at this stage, which uses a direct evaluation of the current plan and possible alternatives in order to decide how to proceed; the different options are fully discussed in the sections on repair and replanning. *Plan Commitment*, which is related to Shadbolt's *decentering*, controls how easily the plans are abandoned for alternatives as the system learns more about the partner's knowledge. Pollack et al. [PIB87] give an architecture for resource-bounded agents where plans are filtered out based on whether they are compatible with current intentions and plans. If the predicted success of an incompatible plan exceeds predicted success of the current plan by more than the setting of a threshold parameter, then the old plan is abandoned. Agents with low thresholds are "bold", and agents with high thresholds are "cautious". Replanning in the JAM system fits this model, and plan commitment is changed by varying the threshold. It is not possible to prove that particular agents are low or high risk with respect to plan commitment, since the routines which agents use for evaluating plans are not visible to outside observers. Examples of agents both repairing and replanning are found in the appropriate sections; these show that sometimes agents choose to repair and sometimes they choose to replan, but they do not explain why.

---

research, since they appear to be due to secondary goals like being friendly or amusing. There is more scope for choice in the unlabelled objects, but these occur so infrequently that agents don't need to build special expressions to pick them out from the other objects.

In our analysis, changing plans "boldly" is a high risk move simply because the model of the partner with respect to the new plan is likely to be less developed and it's easy for the explainer to confuse the explainee by suddenly changing strategies.

The plan commitment parameter is implemented in the JAM system by means of a switch which changes the amount of improvement a replan must make on the current plan before it is adopted. Fuller details of the implementation can be found in section 5.12.4.

3.3 Extension to Other Domains

Although all of the examples which we gave in section 3.2 were from the map task domain, our analysis extends more generally to interactive, collaborative problem solving. In this section, we give examples of behaviours due to some of the communicative posture parameters in two other domains. The first is the assembly of a toy water pump; the examples which we use are excerpted from examples given in Goodman [Goo86]. The second is the explanation of electronic circuit diagrams, with examples excerpted from Cawsey [Caw89a]. Neither of the papers from which we excerpt the examples involve the kind of analysis that we present here. Although we do not cover all of the parameters, the intention is to convince the reader that the analysis transfers as a whole.

We mentioned in section 3.2.1 that the concepts of the electronic circuit domain are the different parts of a circuit and different ways of explaining a circuit's function, such as the "water flowing through a pipe" view or the description using Ohm's law. The *ontology* parameter governs the detail of an explanation by controlling how many and which of these concepts the explainer uses. For instance, an agent who explains a circuit using both styles of description and mentions all of the components would be using low risk ontology. An explainee who asks the explainer about parts of circuit after the explainer seems to have finished uses low risk *ontological resolution*, as in the following extract, where E stands for "expert" and N stands for "novice":

> E: In this circuit we have an output whose voltage depends on the amount of light falling on the LDR and the more light falls on here the more it will conduct so the higher the voltage will be here.
> N: What does LDR stand for?
> E: Light dependent resistor.
> N: What's this bit?
> E: That bit's a variable resistor so you can change how the output level varies with the resistance of the LDR.

Ontology works similarly for the water pump task, although it is possible that there is a smaller range of ontologies which are sensible to use. Since it's clear that the goal of having the

information is to be able to assembly the water pump, agents tend to use all of the pieces and describe them in terms of physical connectivity. However, it would still be possible to vary the level of one's ontological risk in the water pump assembly. For instance, in a low risk setting one might describe subassemblies of several pieces, and especially their functions, since in the long run that might make it easier to assemble the entire pump. Alternatively, high risk agents might not describe all of the physical connections, assuming that some of them are easy to make even without descriptions.

The *difference* parameter doesn't really enter into the water pump domain, since the agents are told that they have exactly the same parts and none of the agents have assembled the water pump before. However, it plays an important role in the circuit diagram domain, since different experts may make radically different decisions about what novices know. As in the map task domain, high risk agents make assumptions about their partners' knowledge which may turn out to be incorrect:

> E: These components here, you might consider them as being both resistors. Two variable resistors. I can write down a relation for resistance...
> N: You'll have to tell me what a resistor is.

Alternatively, low risk agents ask their partners about their knowledge when they are unsure how to proceed:

> E: OK, do you remember anything about transistors?

As in the map task dialogues, if an agent, once he or she has collected information about its partner's knowledge, ignores that information, the agent must have a high risk *partner modelling* setting. In this case, the expert subsequently explains transistors, so he or she must have used the partner model which was constructed, and thus has a low risk partner modelling setting.

Since one would ordinarily expect the water pump explainers to describe the assembly in an order which could be followed in order to put the pump together even if it takes extra effort to construct such an ordering, we can consider the following face-to-face extract, where the explainer gets the ordering wrong, to exhibit high risk *coherence*:

> E: now take the cap base thing and screw it onto the bottom ooops, un-undo the plas- no the clear plastic thing that I told you to put on sorry. And place the little red thing in there first, it fits loosely in there.

Of course, the speaker doesn't intend to be this incoherent; but if he or she had been more careful about planning what to say, the correction would not have been necessary. It is more

difficult for us (as non-experts in electronic circuitry) to assess what makes a coherent explanation of a circuit diagram, although it is clear that some explanations are more orderly than others.

Context articulation and *resolution* operate in both the circuit diagrams and water pump assembly. For instance, low risk context articulation leads to the following meta-comments in a circuit diagram explanation:

> E: What I'm going to do is to get you to explain this last circuit to me. Before I do that I better say briefly what a comparator is.

A meta-comment analogous to the first of these two might be realised in the water pump assembly as, for instance, "Now we're going to put together the main body of the pump". High risk context articulation, as in the map task domain, involves not being explicit about the context of the current section of the dialogue. In high risk cases, then, a partner using low risk context resolution can ask what the current context is and a partner using high risk context resolution can guess or use the first one that comes to mind.

It is difficult to assess *focus articulation* and *resolution* in either of the two domains. In the electronic circuit dialogues, the agents often use deixis, which isn't recorded in the transcripts. Thus it is impossible to tell just on the basis of the transcripts what focus information is being conveyed explicitly and what is left to be inferred. In the water pump domain, either the agents can use deixis (as in the face-to-face trials) or else they don't know where in the visual field each of the parts is on the partner's table, making all objects which have not yet been mentioned in equal focus. However, given access to more information about the trials in which deixis is possible, we could ascribe high or low risk focus articulation and resolution settings to the agents by taking into account whether or not the agents explain and ask about awkward focus shifts.

The *specification* parameter shows up clearly in the non-face-to-face trials of the water pump domain. For instance, one low risk agent uses "clear plastic elbow joint" (excerpt 1) to describe the same piece as what a higher risk agent means by the "clear little attachment" (excerpt 5). The second description is ambiguous since it could mean the pump's stand. Since each of the assembly pieces has a number of distinguishing features, there are many choices for how to describe them and how much detail to use. *Description resolution* works analogously. There isn't as much scope for different specification settings in the electronic circuit domain because agents can be unambiguous by pointing, and, as for the focusing parameters, it is unclear where an agent is being ambiguous and where the verbal description is supplemented by non-verbal information.

Finally, since, as we shall see in section 4.3, repair and replanning both apply more generally than just to the map task domain, the *plan commitment* parameter transfers with them in order to control in what circumstances an agent is predisposed to replan rather than repair.

We have argued in this section that the communicative posture parameters apply in two domains (the explanation of electronic circuit diagrams and the assembly of a toy water pump) which are quite different from the map task domain. We hypothesise that the analysis applies generally in domains in which agents must collaborate interactively in order to solve a problem, since it is these domains in which agents must decide how it is most efficient to bring together information which is not known to any one agent. Special circumstances in specific situations may make certain parameters not apply (for instance, the difference parameter does not apply in the water pump domain because the agents know exactly what objects their partners have), but in general the analysis transfers to other domains.

3.4 Summary

In this chapter we have given a number of parameters along which human participants in the map task vary their strategies, and classified the different possible strategies as low or high risk, depending on whether the strategy involves expending extra effort initially in order to form a careful plan or whether the strategy uses less effort in the first instance and relies on recovery strategies in case of plan failure. This analysis expands upon Shadbolt's analysis of communicative posture parameters by taking the role of the explainee into more serious consideration. It also divides the parameters into subsets according to the part of the agent's planning which they affect and makes the parameters more independent of each other so that they are more amenable to control in a working system. Ideas about future work on the communicative posture parameters are given in section 9.2. Details about the implementation of the ontology, partner modelling, difference, context articulation, and plan commitment parameters are given in section 5.12. This chapter provides a theoretical basis for the behaviours produced by the JAM system and for the strategies which high risk agents in the human corpus use in order to recover from failures, which we will consider next.

Chapter 4

The Recovery Strategies

We saw in the last chapter that there are a number of parameters with which agents can decide either to include information explicitly in a task-oriented dialogue, expending effort to do so, or to assume that the partner will be able to infer the missing information, saving effort in the short term but taking the risk of having to put further effort into recovering from a misunderstanding later on. We have also hypothesised that agents make these choices in such a way as to minimise the total amount of effort that they think they will have to spend on the task. Our treatment of the dialogues so far suffices for both low risk dialogues and high risk dialogues where, fortuitously, the agents understand each other. This chapter analyses what happens in the remaining dialogues, where high risk communicative postures lead to some kind of confusion. The missing element is a set of strategies which the agents can employ to recover from misunderstandings. This chapter gives a set of such strategies, which, following Moore [Moo90], we call "recovery strategies". The analysis given in this chapter will be used as the basis for the implementation of a number of recovery strategies in the JAM system.

The recovery strategies divide into two sets. Section 4.1 discusses recovery strategies which are specific to tasks which involve communication (i.e., multi-agent domains where the agents cooperate to reach a goal). These strategies fall out of the need for the agents to coordinate their planning processes, and so they are not useful for more traditional single agent planning domains. Section 4.2 discusses strategies which are applicable in any system which must be able to recover from plan failure. Section 4.3 argues that the recovery strategies operate more generally than just in the map task domain by giving examples from two other task-oriented dialogue domains. There are nine recovery strategies in all; figure 4.1 lists them and indicates which ones figure prominently in the map task and which ones are implemented in the JAM system at least in some restricted form.

Figure 4.1: The Recovery Strategies

Strategy	In Map Task?	Implemented?
Cooperative Strategies		
Goal Adoption	yes	yes
Ceding Control	yes	no
Elaboration	yes	no
Omission	yes	no
General Purpose Strategies		
Repetition	no	no
Ignoring the Problem	yes	no
Repair	yes	yes
Replanning	yes	yes
Reinstantiation	no	no

This chapter only analyses the recovery strategies in isolation; a discussion of the implementations and of how both past systems and our own agents decide which recovery strategy to use is deferred to chapter 7.

4.1 Communicative Recovery Strategies

The recovery strategies discussed in this section apply in domains in which agents must communicate in order to cooperatively reach a goal. Communication is usually defined in linguistic terms; for instance, as in Grice's definition of non-natural meaning, as given by Levinson [Lev83]:

> S [the speaker] *meant-nn* z by uttering U [an utterance] if and only if
> (i) S intended U to cause some effect z in recipient H [the hearer]
> (ii) S intended (i) to be achieved simply by H recognising that intention (i).

In the map task, all communication fits this definition because it is linguistic; there is a partition dividing the two agents which precludes communication by gestures or by observation of actions. However, in general, the recovery strategies described in this section are not restricted to linguistic communication. Each of them involves an attempt to coordinate the dialogue in some way which prevents or recovers from a breakdown of the dialogue. The first two, goal

adoption and ceding control to the partner, make the dialogue run more smoothly by controlling who takes the initiative of planning the dialogue. Finally, elaboration and omission, which are used to agree upon the level of detail provided and required in any explanations in the dialogue, are used to recover from failures due to the agents have differing notions of success for the task.

4.1.1 Goal Adoption

Goal adoption is a technique by which an agent may indirectly further recovery from a problem in the dialogue by deciding to have as one of her own goals some goal which she recognises the partner as having. Typically such recognition occurs as a result of a dialogue move that the partner has made, and we can describe it intuitively as having the agent "step into the shoes" of the partner and guess what the partner might be doing. Once goal adoption has occurred, then the adopting agent may initiate a recovery strategy to achieve that goal. This is especially useful if the agent can explain to the partner some reason why the partner's initial plan or subsequent attempts at recovery have failed, or if the agent has some other extra information which will make her attempts at recovery more likely to succeed than the partner's. For instance, in our domain, the route giver might have the goal of telling the follower where the swamp is, and the follower might be able to discern this from the description "the swamp is beneath the mountains" even if the follower doesn't understand the description. In this case, even if the follower doesn't have the goal of locating the swamp it makes more sense for her to point out the exact difficulty rather than making the route giver guess what is wrong. It is also common for agents to adopt each other's goals even when plan failure has not occurred just as a way of being cooperative and of being careful about the course of the dialogue.

Since we can not be sure that an agent has adopted the partner's goal in an attempt to recover from a plan failure unless she subsequently uses one of the direct recovery strategies, we can not provide any examples here, but goal adoption will play a role in several examples of other recovery strategies. Quite often it is perfectly possible for the agents to complete the dialogue without using goal adoption at all. An agent who does not goal adopt completes whatever dialogue games are initiated by the partner, answering questions and, in effect, nodding or looking puzzled to the partner's statements, but never expends any effort to tell the other agent what has gone wrong or to suggest alternate plans. This happens especially when the agent has goals of her own which she considers to be more important. We might consider such a participant uncooperative, even though she strictly follows the rules of the game; she simply nevers initiates dialogue that does not serve her own interests. Since the high level goals of the participants in the map task are shared, it is usually in their interests to adopt each other's goals unless they know a better way to fulfill a higher level goal than the one proposed. More

generally, once an agent is committed to participating in a dialogue, she will tend to goal adopt, since such a strategy usually requires less effort and time in the long run. However, exceptions do occur, especially where the partner requests significant amounts of effort or where the agent is otherwise disinclined to be cooperative. Therefore we include goal adoption as a recovery strategy that may be used at the agent's discretion, preparatory to employing some other strategy to actively achieve the goal. A limited version of goal adoption is implemented in the JAM system; the implementation is described in section 7.2.5.

4.1.2 Ceding Control to the Partner

In order to make the dialogue coordination smoother, agents may suspend planning and recovery strategies and allow the partner to control the course of the dialogue. We call this "ceding control to the partner". Task-oriented dialogue is more efficient if the agents concentrate on one goal at a time, and so cooperative agents often allow their partners to keep controlling a dialogue if things are going well. Ceding control to the partner may be used as a strategy at any point in the dialogue, but it is especially useful during a recovery, since it is inefficient for an agent to begin recovery for some goal if the partner is already working on a plausible recovery of that goal, and so agents may cede to the partner in this case. The main exception occurs when an agent has some information which is unknown to the other agent and which makes her believe that the recovery will fail. Another time that agents do not cede to each other is when both have a secondary (and perhaps unconscious) goal to be "dominant" in some sense, and therefore to control the course of the dialogue. In cases where an agent does not have time to determine what problem occurred before the partner attempts a recovery, it may not be possible to tell the difference between an agent which adopts the partner's goals but then cedes control and an agent that does not adopt the partner's goals.

Although it is difficult to tell where an agent has decided to cede the turn and where her partner has just happened to take control first, it is clear when both agents have decided not to cede the turn. For example, in the following extract, agent G seems determined to continue describing the route, while agent F wants to explore the area more thoroughly first:

MAP 11
G: <Draw the line across/
F: I have a cactus. Do you have a cactus?>
G: No. <Draw/
F: Do you have a r-... Sorry. OK.>
G: Draw across to the right and up, round the imaginary obstacle which is halfway between the diamond mine and the edge of the page.

In this example the agents disagree about how to pursue the initial planning for this section of

the route, but in general it is possible for agents to disagree about any of the possible ways of continuing from the current point in the dialogue. Of course, one agent ordinarily eventually cedes control, as F does here; it would be unusual for the agents to talk simultaneously for any length of time. Ceding control is not implemented in the JAM system.

4.1.3 Elaboration

Elaboration is a task-oriented dialogue recovery strategy which is applicable whenever an agent gives an explanation which is not adequately detailed. This kind of failure occurs frequently in the map task dialogues, seemingly because the task is difficult enough that the route giver has trouble imagining how much information the follower must have in order to be able to draw the route on the paper. This may come about as the result of a mismatch in the participants' expectations, as in the following example:

MAP 7
K: Right, so you're at the coastline now, having circled the pagoda.
C: Mhm.
K: And you want to hug the coastline, going further north.
C: <Am I inside the coastline /
K: Oh sorry, yes,
C: or outside?
K: Inside the coast, yeah. >
C: Inside.
K: I just assumed that we weren't sailing or anything.
C: Right.

Here, agent K assumed that even though he did not include the information explicitly, agent C would stay on dry land. This example seems to be a straightforward case of the two agents not totally agreeing on what the task is; K is role-playing, whereas C views the task as drawing on a piece of paper. In other cases, even if the agents view the task similarly, the route giver simply does not give enough information to draw the route:

MAP 1
C: now let it's as it's again as if you were encircling something but I think there's nothing on either of our maps to encircle
M: well I've got an east lake or I can go round the side
C: well you want to still stay above the east lake and do a loop
M: clockwise or anticlockwise
C: clockwise and there's nothing there's nothing in that loop it's a small loop erm it would be about an inch and a half height at its maximum
M: so I go down I've got to go I've got to go
C: and up again
M: down and up with an inch and a half diameter and it's a circle
C: no it's a loop it's it's erm

M: do I go more
C: more oval
M: right with with the
C: but then I'll tell you don't do anything yet you do know the shape that you're gonna get with the loop
M: yes , sort of a pear shape
C: a thin loop exactly exactly
M: but a loop that goes from north to south or east to west
C: that goes from east to west
M: fine

In this case it seems unlikely that C thought that M would be able to use her directions on the first try; she thinks that a high risk posture with plenty of interaction during the recovery is the most efficient way to complete this part of the task. In other cases the route giver may be too overloaded to realise that the missing information is necessary, may think that the partner will be able to infer it, or may simply have a non-exacting definition of success for the task. There are also cases in which the route giver gives a reasonable amount of information about the route, but the follower asks for clarification just to "slow down" the dialogue and force the giver to adopt a lower risk communicative posture. This tends to happen early in the dialogue, when the agents are deciding on the style of the interaction. For instance, consider the following initial extract of a dialogue:

MAP 11
C: OK. Do you have a diamond mine?
D: Yep.
C: If you go from the start
D: Mhm.
C: and go down and to the left of the diamond mine
D: OK.
C: and
D: Straight down?
C: just as you get to the bottom of the diamond mine, stop.
D: Straight down, Chris?
C: Yeah.
D: Ok. Ok, now.
C: Move right across the page.
D: Underneath the words "diamond mine"?
C: Yeah.

In this extract, D is clearly trying to get C to go into more detail about the route. For instance, although in the absence of other directions he could easily have inferred that he should go "straight down", he chose to force C to make that part of the instruction explicit. Technically we will class this move as a request for elaboration, although D may have more global goals in mind than fully understanding this one section of the route.

In all of the examples so far, the elaboration has been requested by the route follower. However, we do not wish to rule out cases where an agent elaborates spontaneously, as in the following extract:

MAP 11
G: Do you have an outlaws' hideout?
F: Yep. It's above me.
G: OK, well, you go up towards it but draw just beneath the words "outlaws' hideout".
F: OK.
G: Is that right over in the right hand side?
F: Yep.
G: OK.

Probably in this case agent G was not satisfied that agent F meant that he understood what to do just on the basis of F's first "OK" response. Explaining why agent G elaborates in this case requires some notion of differing degrees of evidence for a belief, as in Clark and Schaefer's [CS89] work.

To summarise, elaboration is a recovery strategy that is appropriate whenever one agent gives an explanation which is less detailed than the partner requires. This mismatch can be due to a number of factors, including an inaccurate model of the partner, too high a cognitive load, or disagreement about what it means to succeed in the task. In any of these cases, elaboration involves providing more detail in the explanation, either by answering specific points that the partner has raised or by spontaneously offering extra information.

4.1.4 Omission

In all of the cases of recovery by elaboration, one agent gave an explanation which was less detailed than her partner required. It is also possible for the agent to give too much detail. Recovery by omission involves ignoring part of the explanation. We do not include as omission the case where an agent finds the given detail superfluous because she is more expert than the speaker thought; in this case, recovery is not necessary, since the explanation was understood even if it was inefficient. Instead, we consider omission to be a viable option when the second agent decides she can finish the task successfully without having the extra information. It is not always possible to tell from the dialogue transcripts whether an agent has omitted some part of an explanation. However, quite often the agents discuss whether or not the omission is reasonable, as in the following dialogue:

MAP 10
... much discussion about the positions of two cacti...

K: OK + now going back to the first cactus
L: yeah
K: directly above that cactus you have a crashed plane +
L: ah right + erm + oh yes + er + I have a crashed plane marked here + can I + check this + my crashed plane is ABOVE + it's in the BASE of the quadrant + top right hand imaginary quadrant of the + erm + picture + yes er + that SOUNDS too high for me +
K: er
L: + because my cacti + my cacti + are about + oh + I would say + about two inches at the moment or one and a half inches below the + imaginary horizontal + er you know the half way line or the vertical dimension of this + er + piece of paper
K: mm + where we fouled up + I think was + when we + calculated from the mountain
L: OK + mm + yes + so I should shift up my cacti a little
K: OK
L: does does this MATTER really or can we just sort of
K: I, I doubt if they're worried
L: hha!
K: yes
L: OK we'll just .....? so + we both have a crashed plane and it's just above the imaginary horizontal
K: OK + yeah + if you divide the whole page + the whole map + horizontally in half
L: mhm hm
K: your crashed plane is about + an inch above +
L: yeah
K: er the base of your cactuses is about an inch below that
L: yeah + OK + wa- we're near enough

In this extract it appears that the participants' notion of success changes as they discover the task is really harder than they thought. This happens in several of the dialogues, especially if the agents discover that they haven't ended up in the right spot but they aren't sure where they went wrong, as in the following example:

MAP 1
M: you see I don't know where I'm meant to be ending up ok I'll just do what I think you want me to do
C: the point that you want to be at now ok ought to be about two and a half inches from the left margin
M: well I'm nowhere near that erm I'm sort of just above the west lake anyway
C: but have you done that banana curve yet
M: yeah
C: and what point are you at now
M: well I'm pretty much below the banana tree by about three inches erm but I'm you know I'm near I'm far nearer the middle of the page
C: well never mind I mean
M: I've died but go on

However, there are also more localised cases of omission, where it seems likely that the agents could have cleared up the problem without too much difficulty:

MAP 4
K: so you've come in a sort of backward S shape you see it now
H: well I don't see the backward S shape but never mind
K: but you've gone round the top of round rocks and up the top and round the side of apache camp yes
H: yeah

Here, even though H did not totally understand K's explanation, she thinks that what she has drawn suffices. Any case where an agent gives information which is not understood and which the partner ignores counts as recovery by omission in our analysis.

4.2 General Purpose Strategies

All of the strategies presented in the last section were specific to tasks in which two or more agents collaborate. The strategies presented in this section apply more generally to any system which interleaves planning and execution and receives feedback from the real world about the success of its plans. Two of these strategies, repetition and ignoring the problem, involve little effort on the part of the system but yield correspondingly poor results. The remaining strategies, repair, replanning, and reinstantiation, which are also discussed by Moore [Moo90], involve more effort but are more likely to solve the problem.

4.2.1 Repetition

Repetition as a recovery strategy simply involves repeating the action which has failed to produce the desired result. For instance, an agent who is attempting to open an door might turn the handle and push, but fail for some undiagnosed reason. A perfectly reasonable thing to do next would be exactly the same actions again. The effects of real world actions are unpredictable enough that it is quite often worthwhile not to consider whether one's plan is incorrect or whether a precondition (such as unlocking the door) has not been fulfilled until one has tried the original action several times. This is especially true in cases where taking the action is not likely to make one's position any worse by, for instance, damaging the objects involved. Repetition is not very sensible in our domain because by and large people do not understand utterances about simple concepts better if they hear them several times. We found no straightforward instances of repetition in the map task dialogues.

4.2.2 Ignoring the Problem

Another recovery strategy is simply to ignore the problem and hope that it will go away. For instance, if one is building a bookcase and sees that one hasn't nailed a piece on quite straight, one might decide not to do any recovery, but hope that the tolerance for error in the rest of the operation will be enough that the failure won't matter. Ignoring the problem doesn't usually lead to discernible effects on behaviour, and so it is hard to recognise when another agent has chosen this strategy unless one can reason that the other agent must know that the problem exists. Omission is a form of ignoring the problem which only occurs in communicative tasks, since it involves ignoring some detail given in an explanation which is difficult to process or which seems to contradict other more important information. However, omission only applies as a strategy for one particular form of plan failure, whereas ignoring the problem is applicable whenever the failure doesn't make it impossible to continue with execution of the plan. We can't give examples from the human corpus of ignoring the problem in the more general sense because agents don't usually say anything about what they have decided to ignore. Ignoring the problem and ceding the turn can look very similar in the surface level dialogue, but they are distinguished by what happens next after the strategy is chosen. If an agent decides to ignore the problem rather than ceding the turn to the partner in the hopes that the partner will initiate recovery, then the agent will continue with the next part of the plan immediately. Ignoring the problem is not implemented in the JAM system.

4.2.3 Repair

"Repair", following Moore [Moo90], designates any attempt to correct a plan after it has been partly or fully executed by examining the prerequisites of the plan and planning for those which were incorrectly assumed to be fulfilled before the plan's execution was begun. Repair is a particularly important recovery technique in the map task domain. Many of the high risk communicative posture settings involve presupposing a piece of information in the dialogue, and so the dialogue can be recovered by presenting the information belatedly. Since, as we shall see in chapter 7, these communicative posture parameters are handled using optional prerequisites to the planning operators, recovering from them in this way involves repair.

The definition we have given for repair is very low level, since it involves examining the prerequisites to the plan. In this analysis, we assume that knowledge which plays an auxiliary role in the dialogue (e.g., knowledge about the current focus or about objects on the map which must be shared between the agents before the route can be described) is framed in the natural way as plan prerequisites. We classify as repair any recovery which involves belatedly presenting

knowledge that was presupposed when the plan was executed.

The most usual kind of repair present in the human dialogues involves wrongly presupposed knowledge about the existence of an object on the map. For instance, consider the following dialogue extract:

MAP 5
K: Past the volcano and, erm, di-... north of Fast-Flowing River and not at the picket fence.
C: Where's the fast-flowing river?
K: Oh, there's a river on my map. Never mind.
C: Whereabouts is that ?
K: Well, the river is south of the picket fence, mountain and pillars, and there are crocodiles marked above the river. Erm, if you've got the same starting point as me, then the river actually divides the whole page.
C: Oh fine, I've got the river.

In this dialogue, C adopts K's goal for C to understand the main description and then requests the repair specifically. The two agents decide to go along with the repair after initial hesitation on the part of K. If C had chosen not to adopt K's goal, C would have only said "Huh?" and let K proceed with the recovery. Repair initiated by the explainee may be signaled either by a question, as in the extract, or by a statement (such as another agent's "I don't have a well"), as long as the explainee requests presupposed information. We classify any attempt to clear up a presupposition as repair, no matter who initiates the recovery or who diagnoses the problem. This example was due to a high risk difference posture. Of course, other high risk postures, for instance, high risk context articulation, can also lead to repairs:

MAP 9
R: I am going to draw the swamp for you first
G: ok
R: ok um um if you go about two inches across from the top of the tallest tree in the palm beach
G: so I go to the palm beach
R: no don't draw the route just now I am just showing you where the swamp is sorry
G: oh OK

Despite R's telling G the current context before beginning this part of the explanation, G misunderstands and thinks that the current task is to locate the route rather than the swamp. This is probably due to R's rather odd choice of words in his descriptions; he uses the words "draw" and "go" to locate objects, and phrases like "you are going to come about one inch below that palm beach" for the route. R detects the failure and explicitly tells G the context; as the dialogue progresses, R adopts clearer terminology as a precaution. The implementation of repair in the JAM system is restricted to recovering from erroneous presuppositions about the

existence of objects on the map, since these are the only presuppositions which JAM encodes. This encoding is described in section 5.7. Another limitation of the implementation is that agents can only use very simple, predetermined descriptions of map objects, which makes the repairs which they suggest less varied than those which occur in the human corpus. This limitation affects descriptions of the route and map objects for other purposes as well (such as planning what to say in the first place), and is discussed more fully more fully in section 5.1.

4.2.4 Replanning

Again following Moore, we define replanning to be recovery in which the agent abandons the current plan for some goal and begins work on a new plan for it. Replanning can be done for either low level goals or high level ones (in which case, all of the subgoals of the replanned goal which arise only in the current plan are abandoned). In the following example, very shallow replanning is performed:

MAP 12
1G: Right. Start from the sandy shore,
1F: OK.
2G: moving down... straight down.
2F: How far?
3G: Down as far as the bottom of the well.
3F: I don't have a well.
4G: Ah. Right, eh. Move down, eh, vertically down about a quarter of the way down the page. Move to the right in... Do you have local residence?
4F: I do.
5G: Right, well, move up and round and above them.
5F: OK.

In 2F, the route follower starts an elaboration recovery which fails because he doesn't understand the extra detail. In 3F, he signals that failure. 3F can be seen as a request for a repair, since F tells G exactly where the failure is; however, G chooses to replan instead in 4G. Neither agent mentions the well in the remainder of the dialogue.

It is also possible for an agent to replan very early in a plan's execution, even before the main description has occurred:

MAP 9
57M: you cross it ++ erm + and you + do you have a wood + on that + near there + near my + other side of the bridge
57B: <erm + I have a wood + erm + on the far left /
58M: no not that one
58B: hand side of my map no >
...
66M: OK + just below the delta there is a wood but you don't have it

66B: yeah I don't have it there
67M: yeah + so what you + you have to do is do another semicircle + erm + going right from the bridge and then returning back + the semi-circle would be about one inch in diameter

Here, it seems likely that M already had a plan in mind that involved the wood and chose to replan using a different description rather than explain to B where the wood was. Because she used a low risk difference posture to check the reference for the wood ahead of time, her estimate of the effort involved in each of the possible plans changed enough that the plan which didn't involve the wood became more viable. In this extract, not much work had to be abandoned when replanning was chosen; all of the intervening dialogue had to do with the part of the route before the wood. In other cases, much more effort is at stake:

MAP 10
1K: +++ now + skirt fairly closely at about one eighth or one quarter inch down
1L: mhm
2K: the left hand side of the graveyard
2L: mhm
3K: until you come to a line which is the extension of the baseline of the + actual gravestones
3L: right
4K: now + there's a fairly straight + descent + from that point at about a forty-five degree line + down + to the right + it goes er + approximately two inches leaving you about er + oh + about one inch three quarters of an inch away from the old temple
4L: er ++ from the top bottom or + what of the old temple
5K: its only the bottom of the ??
5L: OK
6K: ++ yeah + I'm sorry erm + now + you should + darn + OK lets start again with the graveyard
... a replanned description with the same basic content but more interactive...
11L: OK + but er I mean this brings me into close proximity to my Buddhist temple + I'm just wondering whether it's above below or +
12K: er
12L: what
13K: + er + this leave you about a half inch from ? + topmost part of the temple
13L: OK
14K: which is to your right
14L: OK right below the elephant
15K: yes + right

Here K decides to replan rather than elaborate based on L's question in 4L, giving a more interactive description, presumably because he thinks that the recovery will be more complicated than L does. Unfortunately, L has exactly the same problem with the second description.

In the examples, replanning has been tried as the first recovery strategy after a recognised failure. However, replanning can occur at any time when the agent decides that another plan

would be better, even if no problem has yet become evident, or if another recovery strategy has already been tried. The choice depends only on the current estimate of the amount of effort that remains to reach the goal given each of the possible ways of continuing the dialogue. Information about how replanning is implemented and about how a replan in chosen in the JAM system is given in chapter 7.

4.2.5 Reinstantiation

In Moore's terminology, "reinstantiation" is an attempt to correct a plan by changing the values of the variables in the plan (for instance, by replacing a bent nail which led to a plan failure with a straight one) and running it again, and "replanning" involves replacing the current plan with a totally new plan for the same goal. Using Moore's definitions, there is at least a sense in which reinstantiation and replanning are equivalent operations. As long as there is only a finite set of possible instantiations for any variable in the plan, any reinstantiation can be rewritten as replanning by creating a possible plan for each possible instantiation. Conversely, it is always possible to express the replanning of a goal as reinstantiation if one creates a new variable whose constraints indicate that it can only be bound to one of the possible plans. Depending on how the system is implemented, it may not even make a difference to the planner's performance which way the planning operators are written. However, intuitively there is a difference between the kind of information which is expressed in variables and that which is expressed as complete plans which involves the amount of commitment to the choice of domain objects. As in the "nail" example, variables should be reserved for domain objects which are, in some sense, trivial, whereas complete plans should have a more substantial nature. Then we can differentiate between reinstantiation as a fairly simple recovery strategy and replanning as a more drastic measure (although, as we shall see in chapter 7, this does not imply that we will wish to apply reinstantiation before replanning whenever possible). We therefore accept Moore's definitions of reinstantiation and replanning, with the proviso that the plan operators be formulated so that variables only refer to easily replaceable domain objects. There are no easily replaceable objects in the map domain, and so reinstantiation as defined here does not occur. Unfortunately, JAM plan operators, as discussed in section 5.7, use variables more generally to refer to, for instance, entire descriptions of the location of an object. We consider reinstantiating these variables to be a form of replanning, and so our implementation of replanning uses a combination of standard replanning and variable reinstantiation. This point is discussed further in section 7.2.3.

4.3 Extension to Other Domains

In this section, we argue that the recovery strategies which we have discussed with respect to the map task also apply in other domains. We have already argued that the general purpose recovery strategies apply in non-collaborative domains, primarily by reference to the task of building a bookcase. It remains for us to argue that both the communicative and the general strategies apply in other interactive, collaborative domains. For this purpose, we give examples from the water pump assembly and the electronic circuit domain, both of which were introduced in section 3.3. As in that section, we do not cover all of the strategies but our intention is to convince the reader that our analysis of recovery strategies transfers as a whole.

Goal adoption and *ceding control to the partner* apply in any domains where two or more agents have the same general aims but may have different ways of fulfilling the common goal. For instance, suppose that in the electronic circuit domain, the novice, unbeknownst to the expert, knows something about Ohm's law. Both agents have the goal that the novice understand the circuit. If the expert begins by explaining the circuit in a new way, then the novice may adopt the expert's goal and try to learn that way of understanding the circuit. Alternatively, if the novice interrupts the expert, the expert may cede control to the novice and eventually learn that the most efficient course of action is to replan and use Ohm's law for the explanation. There is less possibility for goal adoption and ceding control in the water pump assembly because the experiment is set up so that one agent has all of the relevant knowledge and the other has none, although situations where non-face-to-face agents must correct an assembly which the explainee has put together incorrectly may involve either of the strategies.

Elaboration can occur in either of the domains, since the explainer may provide an explanation which is not detailed enough. For instance, in Goodman's first excerpt the explainee asks for more information about one part of the water pump assembly:

> E: And put it [the clear plastic elbow joint] over the bottom opening, too.
> N: Okay.
> E: Okay. Now, take the–
> N: Which end am I supposed to put it over? Do you know?

In one of the circuit examples, it takes the novice two tries to extract an adequately elaborate explanation of how one part of the circuit works, since the expert stops after explaining the individual components and the overall circuit function but not how the bits go together to provide the function:

> N: Does it matter where these things are?
> E: Yes, if you switch them round it will work the other way round, and the voltage will go down as the light increases.

N: Why?
E: Because, the ratio, the output will vary between 0 and 9V in proportion to the ratio of the two resistances, so the bigger the resistance here, the further it will be away from 0V.

Examples of the *omission* strategy work similarly to the examples of elaboration; in either domain, if an explanation is so detailed that it is confusing, agents may use omit part of the detail.

The *repetition* strategy tends not to occur in these domains for the same reason that it tends not to occur in the map task dialogues; adults seldom think that (barring channel problems) repeating the same utterance will make the hearer any more likely to understand. However, agents do sometimes *ignore the problem* in these domains. As in the map task domain, ignoring a problem in the explanation of circuit diagrams usually only involves ignoring some bit of information that has been exchanged, and can be classified as omission. Other kinds of ignoring the problem occur in the water pump assembly, since one can imagine putting the water pump together incorrectly and never rectifying the mistake. In Goodman's first excerpt, at one point the explainee twists one piece to force it onto the wrong piece. Goodman describes this effort by saying that "people do not always give up when a speaker's description isn't perfect (or isn't readily assimilable for them)", but that "they try to plow ahead anyway" (page 275). If this is the case, then these agents are ignoring the problem.

Repair occurs in the electronic circuit domain where the expert has a high risk difference setting; for instance, in the example from section 3.3, the novice initiates a repair:

E: These components here, you might consider them as being both resistors. Two variable resistors. I can write down a relation for resistance...
N: You'll have to tell me what a resistor is.

Replanning also occurs in the circuit domain:

E: That bit's a variable resistor so you can change how the output level varies with the resistance of the LDR.
N: Say that again.
E: Well, this is a potential divider here, so the potential here will be between zero and 9V in proportion to the resistance of these two things.

The novice may or may not have been asking the expert to repeat exactly the same explanation; if the novice was, then it was a request for the repetition recovery strategy, which the expert ignored.

We have argued in this section that the recovery strategies apply in two further domains, the explanation of electronic circuit diagrams and the assembly of a toy water pump, which

are quite different from the map task domain. These examples are intended to convince the reader that the recovery strategies operate more widely in domains which involve collaborative, interactive problem solving.

4.4 Summary

In this chapter we have given a number of different strategies that agents can use to recover from problems that arise in the dialogue. Some of the strategies are only useful in domains with more than one agent, while others are useful for any system which must recover from plan failures. In addition to defining the strategies, we have also given examples of them from the map task corpus, where such examples exist. These strategies provide a partial answer to the question of what a high risk agent can do when her plan fails. Implementations of some of the recovery strategies and an architecture which allows the agent to choose flexibly among them are described in chapter 7.

Chapter 5

The JAM System

This chapter describes aspects of the JAM system which are essential to understanding how the system operates but which do not in themselves make substantial contributions to the state of the art. Although this chapter does not describe any novel aspects of the system except how communicative posture is implemented, it contains important information about aspects of the system which limit its function, and thus provides the context in which the system output given in remaining chapters should be interpreted. Since our agents can not use the same kind of visual maps that were used in gathering the human corpus, section 5.1 explains exactly what sort of domain knowledge our agents do use. Section 5.2 describes the databases from which the agents receive their information about the domain. Section 5.3 describes the form of goals used in the simulation; section 5.4 describes the use of dialogue games and the language which JAM agents use for communicating with each other as well as how that language is transformed into something vaguely resembling English for the human observer. Section 5.5 describes how plans are represented and how they are executed incrementally so that they also contain a record of the past dialogue. Section 5.6 explains the planning language which JAM agents use and section 5.7 gives their planning operators. Section 5.8 gives the method by which JAM agents use the plan operators to plan. Section 5.9 describes how agents keep track of the cost or effort of the current plan so that they can minimise it and thus adhere to the Principle or Parsimony. Section 5.10 explains how the top level of the system controls the dialogue by swapping between the two agents involved and discusses the limitations of such an approach. Section 5.11 gives an outline of what each agent does whenever it is given the chance to "speak"; more details are provided in sections 6.4.3 and 7.4. Section 5.12 describes how communicative posture is implemented. Finally, section 5.13 summarises the implications of the JAM system's implementation for the human behaviours that it will be able to simulate.

Many of the aspects of the system described in this chapter can be considered to be limitations on the system's operation because the rather simple methods which have been implemented unnecessarily constrain the agents' behaviour. However, the implementations described suffice for the purposes of demonstrating that communicative posture and recovery strategies can be incorporated into a working system. The parts of the JAM system which form major contributions, the belief model and the layered architecture, are described in chapter 6 and 7, respectively.

5.1 The Domain Knowledge Representation

JAM agents have a very simple representation for their maps. All of the JAM dialogues are about the same pair of maps (map 9 of appendix A), although any map described in the correct format could be used. Since human agents don't ordinarily describe the route in one piece, the route must be divided into explainable sections; the route on map 9 is rigidly divided into four sections for the purposes of the simulation.[1] JAM agents represent the map in terms of CONCEPTS which occur in the map domain. Throughout this thesis, concept names will be given in small capital letters. Concepts come in four basic types:

Map Objects are features which appear on at least one of the two maps. In the agent's internal "language of thought", the names of map objects end in digits. Where possible the names are constructed in order to make them easy for the human observer to recognise; for example, some of the route follower's map objects are CASTLE-0, WOOD-0, and WOOD-1. Occasionally an agent may have to represent a map object which is not on its own map, but which appears on the partner's; in this case the agent will create some unique name for the new object such as T114.

Route Objects are the entire route and its four sections. These objects can be identified by their special names, which are ROUTE, SECTION-ONE, SECTION-TWO, SECTION-THREE, and SECTION-FOUR.

Descriptors are concepts which express relative directions such as "left" and "down". The system has available to it a fixed set of descriptors and agents recognise them because they have special reserved names which resemble their English translations. Each descriptor has an arity associated with it, which defines the number of map objects which it must take in order to produce a description. For instance, some of the descriptors used in the

[1] This division does not constitute a major restriction for the simulation. It would not be difficult to allow the route to be divided into an arbitrary number of segments if a procedure for doing so were available.

current system are BETWEEN/2, LEFT-OF/1, ABOVE/1, and TO/1.

Propositions are concepts which describe either the path of one of the sections of the route or the location of a map object. They can be identified because they are lists where the first element is the keyword `'describes`, the second is the name of a map object or one of the sections of the route, and the third is a *description*. Descriptions are built up out of *descriptors* and map objects. A description is simply a list in which the first element is a descriptor, the length of the list is the descriptor's arity plus one, and the remaining elements are map objects. For example, two common propositions are (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0)) and (DESCRIBES SWAMP-0 (RIGHT-OF PALM-BEACH-0)).

The use of concepts within the agent's "mind" is a good way of representing the maps, because it allows the agent to think in terms of getting map and route objects to be shared with its partner by using the dialogue to share with the partner propositions which describe the concepts. However, the way in which the agents derive this representation is too simplistic to do anything but provide rigid descriptions for each of the map and route concepts. Rather than having a representation which reflects the way the map is laid out graphically and reasoning from it to produce descriptions for the dialogue, JAM agents are initialised with a preprogrammed set of descriptions for each of the objects with which they are familiar, and they can not use any descriptions but these. For instance, route givers know exactly three descriptions for the concept SECTION-ONE: (AROUND SWAMP-0), (LEFT-OF PALM-BEACH-0) (BETWEEN PALM-BEACH-0 SWAMP-0). Thus there are only three distinct propositions which a route giver can construct when it wishes to convey the path of the first section to a route follower. The descriptions for other map and route objects are similarly restrictive. Agents can not do even the most simple kinds of spatial reasoning in order to construct new descriptions; such an extension would be interesting but is left to further work. In addition, agents are automatically assumed to know what each of the descriptors means in any context, and thus, for instance, the agents can not have a dialogue about what it means to go "between" the palm beach and the swamp as long as they understand the locations of both of the map objects.

As a result of this representation for the map, the only distinction that JAM agents can make about their knowledge of a concept has to do with whether or not they know and understand the components of at least one proposition which describes the concept. Thus JAM agents can not distinguish among whether an agent has superficial knowledge, vivid knowledge or expert knowledge of a concept, as these degrees of acquaintance were defined in section 2.5.2. JAM agents only two degrees of acquaintance, "vividness" and "awareness". They define an agent to hold a concept vivid if the agent knows at least one proposition describing the concept and

holds vivid the map objects used in the description, and an agent to be aware of the concept otherwise; JAM agents do not distinguish among agents that are unaware of a concept and those that are aware of a concept but do not hold it vivid.

The map representation described in this section seriously restricts the simulation because it restricts the communicative posture parameters and recovery strategies which can be implemented. Ontological resolution relies on an agent using spatial reasoning in order to determine what map objects are relevant to a description and have been omitted. Focus articulation and resolution also rely on spatial reasoning to some extent, as does the aspect of specification and description resolution which involves knowing when a description is unambiguous. The elaboration and omission strategies rely on an agent being able to disagree about the meaning of descriptors like "around" and "left". However, the current representation suffices for the purposes of this thesis, and any limitations introduced by it are not inherent in the system but could be improved with further work. In addition, there are other communicative posture parameters and recovery strategies described in chapters 3 and 4 which could be implemented without changing the current representation.

5.2 The Map Databases

In the last section we explained that each agent has a preprogrammed set of descriptions for each of its map and route objects, with which it is initialised before a dialogue begins. These descriptions are read in from an ordinary database. The database contains one entry for every map or route object and descriptor, keyed on the agent's internal names for them. Each database entry includes a simple referring expression for the concept (as we shall define in section 5.4), information about whether the object is known to the partner, not known to the partner, or uncertain, such as would be evident to human participants from the experimental setup,[2] and, if the concept is not a descriptor, the set of descriptions of it. In addition, each entry contains information about how to place the concept into a network of concepts representing the domain which will be introduced in section 6.4.2.

5.3 Goals

Just as in other planners, JAM agents decide upon actions by having goals. JAM goals have three parts: the type of the goal (which is always "KNOW", since JAM agents only desire to

[2]The route objects are marked as being vivid to the route giver but not the route follower. The map objects are marked as being uncertain. Descriptors are marked as being vivid to both agents.

exchange knowledge with the partner), the agent for whom the knowledge is desired (which can be either the agent itself or the partner), and the concept involved. Throughout this thesis, goals will be represented as lists and given in small capital letters, as in the goal (KNOW JOHN ROUTE). Any concept can be a part of a goal. In addition, templates which unify with possible concept names can be part of a goal structure. In this case, fulfilling the goal constructed by instantiating the template in any way permitted by the language of the domain representation counts as fulfilling the goal itself. For instance, the goal (KNOW JOHN (DESCRIBES SECTION-ONE ?S)) is fulfilled if john gets to know any one of the permissible descriptions of the first section of the route.

5.4 Communication and Dialogue Games

JAM dialogue agents communicate by placing "messages" in each other's mailboxes which are identified as particular moves in a "dialogue game". Each message contains the name of the sender, the "type" or game, the role of the sending agent (initiator of the game or responder), the concept involved in the move, and a unique identifier which the agents use to know which moves link together to form a single game. We have reviewed several possible approaches to dialogue games in section 2.2; JAM's most closely resemble Houghton's "interaction frames", since, as we shall explain in chapter 6, they specify belief updates which can be made as a result of hearing particular moves in the dialogue. The JAM system defines the following dialogue games. In the notation given, the first subscript indicates whether the move is an initiation (I), response (R), or feedback (F), and the second subscript, where it exists, indicates whether the move is in some sense "positive" (P) or "negative" (N). Phrases in italics give English glosses of some of the moves.

INFORM Tell the partner the proposition in the proposition slot of the initiating message. This game has two moves: the initial informing move and a "positive" or "negative" reply from the partner:

I_I – initiate an INFORM game

I_{RP} – reply positively to an INFORM initiation (*ok*)

I_{RN} – reply negatively to an INFORM initiation (*I don't understand*)

WH-ASK Ask the partner a wh-question (i.e., ask the partner to tell you a proposition which unifies with the form given in the proposition slot of the initiating message). This game has either two or three moves. The game always begins with the initiating wh-question:

W_I – initiate a WH-ASK game

If the partner knows an answer to the question, then there are two subsequent moves: a "positive" reply (the answer) and feedback which is "positive" if the agent understands the reply and "negative" otherwise:

W_{RP} – reply positively to a WH-ASK initiation
W_{FP} – feedback positively to a WH-ASK positive reply (*ok*)
W_{FN} – feedback negatively to a WH-ASK positive reply (*I don't understand*)

If the partner does not know the answer to the initiating wh-question, then there is only one subsequent move: a "negative" reply which indicates that the partner does not know any propositions which answer the question:

W_{RN} – reply negatively to a WH-ASK initiation (*I don't know*)

YES-NO Ask the partner whether or not it understands the concept given in the proposition slot of the initiating message. This game is really only a restricted version of what we would consider to be a yes-no question in natural dialogue, since JAM agents never need to ask such questions except to establish information about their partners' knowledge. This game has two moves: the initiating question and the answer, which is "positive" if the partner understands the concept in question and "negative" otherwise:

Y_I – initiate a YES-NO game (*do you have the X?*)
Y_{RP} – reply positively to a YES-NO initiation (*yes*)
Y_{RN} – reply negatively to a YES-NO initiation (*no*)

OPEN Open a dialogue about the concept named in the proposition slot of the initiating message. This serves to inform the partner about the current context. This game has two moves: the initiating statement and a reply which is "positive" if the responding agent agrees to enter the dialogue about the concept and "negative" otherwise:

O_I – initiate an OPEN game (*I want to talk about the X*)
O_{RP} – reply positively to an OPEN initiation (*ok*)
O_{RN}[3] – reply negatively to an OPEN initiation (*I don't want to talk about it*)

CLOSE Tell the partner that you believe that the concept named in the proposition slot of the initiating message is mutually understood. This game has two moves: the initiating statement and a reply which is "positive" if the partner agrees that the concept is mutually understood and "negative" otherwise. Closing and opening games do not occur symmetrically in the dialogues because agents use them for different reasons. Closing games are used after recoveries in order to coordinate the agents' knowledge.

[3]No agents in the current version of the JAM system ever choose to make the O_{RN} move.

C_I – initiate an CLOSE game

C_{RP} – reply positively to an CLOSE initiation (*I agree*)

C_{RN} – reply negatively to an CLOSE initiation (*I disagree*)

This set of dialogue games differs from Houghton's set of interaction frames in a number of ways. We split his frame for "finding something out" into two parts, one for yes-no questions and the other for wh-questions. This is a useful division because in our analysis, wh-questions initiate games which contain three moves instead of the two in a yes-no game. Our set does not distinguish between "making something known" and "getting something done" by communicating a need to the partner, since we consider the latter to be a subset of the former and because the nature of the map task does not make it fruitful to divide them. However, we have found it useful to distinguish opening and closing games from the more general informing games because they are used under very specific circumstances.

Propositions in the messages which convey dialogues moves are expressed in the same basic language as the agents use for planning, with one difference: agents substitute referring expressions for the internal names in the proposition before they send the message. For simplicity, in the current version of the JAM system, agents already have one referring expression for each concept for which they have an internal name, and if two agents begin the task with the same concept (albeit with different internal names), they use the same referring expression. When an agent hears the partner refer to a novel concept, then the agent simply creates a unique internal name for the concept and adopts the referring expression which was used by the partner. This ensures that agents can not disagree about referring expressions, which, although it is clearly unrealistic, is necessary because the agents can not use spatial reasoning to disambiguate expressions. Improving the use of referring expressions is left to further work. Referring expressions in the current system occur between double quotation marks and are the labels given on the maps or some obvious other form. For instance, the referring expression for what one agent knows as PALM-BEACH-0 is "palm beach". Key words in the concept naming language (such as DESCRIBES) are used directly in the communication language. Then if an agent plans to inform the partner of what it thinks of in its concept naming language as the proposition (DESCRIBES PALM-BEACH-0 (RIGHT-OF SWAMP-0)), it will use the communication language proposition (DESCRIBES "PALM BEACH" ("RIGHT OF" "SWAMP")).

For notational convenience, when we need to specify which agent is making a dialogue move we will prefix the name of the agent to the move, separated by a colon, and when we need to specify which concept (or proposition) is central to the move, we will append the internal name of the concept, from the perspective of the agent whose process we are explaining, in parentheses. Therefore, from B's perspective, the move $A : I_I$((DESCRIBES SECTION-ONE (BETWEEN PALM-

BEACH-0 T114))) is a move by agent A to initiate an inform game concerning a particular description of what agent B knows as section-one (the first section of the route). This notation specifies the name of the concept in the agent's internal language; in some cases, for ease of exposition we will specify the name of the concept in the language of communication instead.

We should stress that in this thesis, we intend dialogue moves to convey the speaker's **intention** but to make no claims about the surface level form of the utterance. For instance, the move $A : I_I$((DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))) could be realised as "The first section of the route goes between the palm beach and the swamp", "Did you know that the first section of the route goes between the palm beach and the swamp?", or even, in the appropriate context, "Go between the palm beach and the swamp." We would not even wish to claim that all dialogue moves are conveyed linguistically; for instance, the I_{RP} move might be conveyed by the phrase "ok", but also by "uh hhh", a grunt, and, in domains which allow face-to-face contact, a nod, gesture, variation in eye contact, or the like. In particular I_{RN} and W_{FN} moves are sometimes realised in human dialogue by no action at all. We also want to allow for the possibility that dialogue moves may be elided in certain contexts. For instance, we would want to represent the dialogue extract

A: Go around the swamp.
B: Where's the swamp?

as consisting of three moves: $A : I_I$, $B : I_{RN}$, and $B : W_I$. However, agent B leaves A to infer the I_{RN} move from the fact that B has asked A a question about knowledge which A has presupposed. We might also think of this situation as one in which B realises two dialogue moves by means of a single surface level utterance, since B's utterance is what allows A to infer the move which is not directly realised. In any case, we claim that even though our dialogue move analysis may seem cumbersome, the dialogue games map very flexibly onto surface level behaviours and really do account for the constructions which are used in the human dialogues.

Finally, we must issue a caveat about the form which the output of the JAM system takes. The agents actually communicate as described above, but a more readable form of output can be included in traces of the system. For instance, the following is a fairly common move in the examples in chapter 8:

mary makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*mary says to john: the first section of the route goes between the palm beach and the swamp.*

The JAM system produces the "translations" which are given in italics *only* for the benefit of human readers. Because the translations are not used for any other purpose, the system

uses extremely unsophisticated templates to construct them and only ensures that they are readable, not that there are grammatically correct. In addition, none of the flexible mapping techniques described earlier in this section are used. Generating reasonable pseudo-natural language output and making the agents understand it is beyond the scope of this research.

5.5 Plans

As is usually the case with hierarchical planners, an agent's plan is represented as a tree. In the JAM system, the top level goal of the agent is at the top of the tree. Any node which is not a leaf node contains a goal, and the children of that node constitute a plan for the goal. Leaf nodes may be either goals, in which case they are ones for which a plan has not yet been formulated or which are satisfied trivially, or games to be executed. The JAM system interleaves planning and execution in the usual way (as in, for instance, Ambros-Ingerson [AI86]) by moving a pointer through the plan, expanding goals nodes, and executing games in a depth first, left to right manner; we shall term this pointer the *currency pointer*. Thus the plan tree serves both as a plan and a model of the past discourse; all nodes which are visited before the node on which the currency pointer rests except its ancestors are part of the dialogue history, the current node and its ancestors are currently under consideration, and the nodes yet to be visited are the future plan. We defer discussion of how an agent adds nodes to the plan to section 5.8. The JAM system marks each plan node with a "status" that indicates what has been done so far with the node. Nodes which contain dialogue games can have the following statuses:

Unexecuted: None of the game's moves have been made yet.

Initiated: The initiating move has been made, but no further moves.

Replied: The responding move has been made, but the game still requires a feedback move. In the current system, this status can only occur for nodes containing WH-ASK games, since all other games only require two moves.

Executed: All of the game's moves have been made.

Nodes which contain goals can have the following statuses:

Unplanned: No plan has yet been formulated for the goal. This can only be true if the node is a leaf node.

Open: The goal has children which constitute a plan for it, but the goal has not yet been satisfied. In this case, the currency pointer is on the node or on one of the descendants of the node, and either the plan has not been completely executed and may not be completely formulated yet, or the plan has been completely executed and has failed.

Abandoned: The goal has been abandoned because one of its ancestors has been replanned.

Succeeded: The goal has been satisfied. The currency pointer will be somewhere after the goal in the plan. A goal may be satisfied without all of its children being satisfied or executed if some plan of the partner's has brought about the goal or if some of its descendant goals have been abandoned due to replanning.

These statuses are updated as the dialogue progresses. Given these node status definitions, the system does not actually need the currency pointer because it can be found by following down a chain of open goals from the top of the tree to either an unplanned goal or an incompleted game. Therefore it is kept purely as an efficiency measure.

Throughout this thesis, trees will be drawn by reflecting the depth of node using indentation. For instance, figure 5.1 contains an example of a plan tree part way through a dialogue between two agents named anne and andrew, from anne's perspective. Anne's top level goal is (KNOW ANDREW ROUTE), and that goal has four children, one corresponding for each of the four sections of the route. The currency pointer rests on the unexecuted inform game.

Agents must keep their currency pointers up to date as the dialogue progresses. In the JAM system, agents consider updating their pointer between every decision in the dialogue of what to do (that is, each time they decide whether to plan, execute, or use one of the recovery strategies next). Agents have three rules for updating their pointers, and they apply all three rules in any order until the pointer does not move:

1. If the status of the current node is 'succeeded, 'abandoned, or 'executed, then the agent moves the currency pointer to rest on that node's parent.
2. If the current node is a goal node which the agent believes is now fulfilled, and if there are no children of the node which are unfinished games (i.e. every child of the node is either a goal or a game with status 'executed), then the agent moves the currency pointer to rest on that node's parent.
3. If the current node has a child which is not yet finished (i.e., a goal with status 'unplanned or 'open or a game with status 'unexecuted[4]), then the agent moves the currency pointer to rest on the leftmost child of the node which is not yet finished.

[4] The case where the child is a game with status 'initiated, or 'replied does not arise.

Figure 5.1: A Plan Tree

```
open: (know andrew route)
   succeeded: (know andrew section-one)
      abandoned: (know andrew (describes section-one
                                    (between palm-beach-0 swamp-0)))
         executed: (12 inform anne andrew
                       (describes section-one
                          (between palm-beach-0 swamp-0))
                       not-ok ?var33)
      succeeded: (know andrew (describes section-one
                                    (left-of palm-beach-0)))
         executed: (26 inform anne andrew
                       (describes section-one (left-of palm-beach-0))
                       ok ?var39)
   open: (know andrew section-two)
      open: (know andrew (describes section-two
                             (between waterfall-0 cliffs-0)))
         unexecuted: (35 inform anne andrew
                         (describes section-two
                            (between waterfall-0 cliffs-0))
                         ?var42 ?var43)
   unplanned: (know andrew section-three)
   unplanned: (know andrew section-four)
```

These three rules keep the currency pointer on the first unfinished node in a depth-first, left-to-right traversal of the plan.

So far, we have omitted discussion of how an agent incorporates its partner's actions into its dialogue history and plan. The partner's responses to games initiated by the agent are simply stored in the games, since rules one and two entail that the currency pointer remains on a game until it is completely executed. If the partner initiates a dialogue game, then the agent pushes it as a child of the current node, unless the current node is a game itself, in which case the agent will push it as the next sibling of the current node in order to preserve the requirement that games must only occur on leaf nodes.[5] This reflects the fact JAM agents rather single mindedly assume that any games initiated by the partner are relevant to the current goal, since the currency pointer update rules require that game to be completed before the current goal can be considered to be completed. As with most of the details in this chapter, we do not assume that this is the best way to record the dialogue history and control traversal of the plan, but it is a way which is simple to implement and which suffices for the current purpose.

5.6 The Planning Language

The JAM domain plan operators have the same basic features as plan operators in a traditional hierarchical planner, in that they have effects, operations, prerequisites, and constraints. The effect is a goal which will ordinarily be brought about by an application of the operator. The operations are the goals which must be satisfied or games which must be executed in order to apply the operator, and the prerequisites are conditions which must be satisfied before the operations can be begun. These are the standard definitions for these parts of a plan operator. There is less accord about the role of constraints in the literature; in the JAM system, constraints are conditions which must hold in order for the operator to be applied, and if they are not already fulfilled when the agent is planning then the agent will not try to satisfy them by taking preliminary actions. This makes the constraints different from prerequisites, since prerequisites are conditions on which the agent will work in order to bring them about. The JAM system allows for two kinds of constraints: nonrelaxable ones must be satisfied under any condition, and relaxable ones can be ignored if there is no other plan operator which has the desired effect. In addition to these standard components of a plan operator, the JAM system adds a replanning flag which indicates whether or not agents should consider replanning goals which have been satisfied using the operator.

[5] This requirement exists because the children of a node are considered to be a plan for that node, and it does not make sense to have plans for executable nodes.

In the JAM system, the effect of an operator must be a goal and the operations must be a list, of which every element must be either a goal or a game. Constraints and prerequisites are more complex because they have their own internal structures.

5.6.1 Constraints

Each domain plan operator has an ordered list of nonrelaxable constraints which must be satisfied in order for the operator to be applied, and another ordered list of relaxable constraints which should be satisfied. Operators for which not all of the relaxable constraints are satisfied are only used if there are no other planning options. When an agent comes to determine whether or not the constraints are satisfied, it works through the constraints one by one, starting with the nonrelaxable ones, building up lists of possible variable bindings which satisfy the constraints. The constraints themselves are represented simply as lists and come in a number of different types, which are identified by the constraint's first element:

SET: For each possible binding, extend it so that the variable given as the second element of the list is set to the value of the arbitrary lisp expression constructed by instantiating the third element of the list with the possible binding. For instance, for any existing binding the constraint (SET ?I (NEW-FRAME-ID))) will add for ?I the value of the expression (NEW-FRAME-ID), which is a unique reference number. This type of constraint does not check whether or not the variable has already been set in the possible binding.

TYPE: For each possible binding, if the variable given as the second element of the list is not already bound, extend the binding so that the variable is set to any object of the type given. In the current system, the only allowable type is 'agent, and the possible values of this type are the two agents involved in the dialogue. For instance, if the variable ?F is not already set in any of the possible bindings, the constraint (TYPE-OF ?F AGENT) will double the number of possible bindings by creating two new bindings for each old one, one with ?F set to one of the agents and one with ?F set to the other.

BELIEF: In this type of constraint, the second element of the list must be a variable which is already bound to an agent and the third element of the list must be a complete or incomplete proposition (that is, one in which there may be unbound variables). If the proposition is complete, then the constraint eliminates any of the possible bindings for which the agent running the constraint does not believe that the named agent believes the instantiated proposition. If not all of the variables in the proposition are bound, then the constraint creates a new list of possible bindings which extends the old bindings in such a way as to set any unbound variables to values which the agent running the

constraint believes makes the proposition true for the named agent. For instance, if john is running a constraint set, and in all possible bindings so far ?G is set to mary, ?S is set to SECTION-ONE, and ?D is not set, and if john believes that mary believes both (DESCRIBES SECTION-ONE (LEFT-OF PALM-BEACH-0)) and (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 T114)), then the constraint (BELIEVES ?G (DESCRIBES ?S ?D)) doubles the number of possible bindings by extending each one in two ways: one so that ?D is set to (LEFT-OF PALM-BEACH-0), and the other so that ?D is set to (BETWEEN PALM-BEACH-0 T114).

ARBITRARY: In this type of constraint, the result of instantiating the entire constraint with a possible binding constructs an arbitrary lisp expression. The constraint eliminates any bindings which do not satisfy that lisp expression. For instance, the constraint (NOT (EQUAL ?F ?G)) eliminates all possible bindings in which ?F is set to the same value as ?G.

Of course, this is not a very sophisticated constraint language, and it is not as powerful as, for instance, Stefik's way of dealing with constraints [Ste80], but it suffices for the current purpose. An example of our constraints in operation is given in section 5.8.

5.6.2 Prerequisites

A prerequisite has two parts: the **applies-test** and the **content**. The **applies-test** is a form which, when instantiated by one of the possible bindings given by the constraints, yields a lisp lambda expression of one variable. The prerequisite is said to apply if applying that lambda expression to the planning agent's communicative posture (a structured object) returns t. For instance, a prerequisite with the applies-test (LAMBDA (PARAMS) (LOW-RISK? (CP-DIFFERENCE PARAMS))) applies if and only if the agent has a low risk difference setting. As we shall explain in section 5.12, the applies-tests in the current system are used primarily for selecting and deselecting prerequisites on the basis of communicative posture. For prerequisites which are universal, the lambda expression returns t no matter what. The **content** of a prerequisite can be a goal, a game, or a "forall" form. If it is a goal or game and the applies-test returns t, then that goal or game is a prerequisite of the plan operator's operations. Forall forms are used to expand out a number of prerequisites from one template. Forall forms are lists where the first element is the token 'forall, the second element is a variable, the third element is the token 'in, the fourth element is a form which, when instantiated with one of the bindings given by the constraints, yields an arbitrary lisp form which returns a set of objects, and the fifth element is a goal or game. If the applies-test of the prerequisite returns t, then the

variable is set to each element in the given set of objects in turn, and a goal or game is returned for each one. For instance, if ?D is set to (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))) and ?F is set to john in one possible binding, the forall form (FORALL ?N IN (OBJECTS-IN-PROP '?D) (KNOW ?F ?N)))) will return two prerequisite goals, (KNOW JOHN PALM-BEACH-0) and (KNOW JOHN SWAMP-0).

5.7 The Planning Operators

The JAM system uses the following planning operators. The names by which we shall denote the operators are given in **bold face** throughout the thesis.

Route: To get some agent to know the route on the map, get some other agent to describe all the sections of it in order.

Effect: (KNOW ?A ROUTE)

Operations:
(KNOW ?A SECTION-ONE),
(KNOW ?A SECTION-TWO),
(KNOW ?A SECTION-THREE),
(KNOW ?A SECTION-FOUR)

Nonrelaxable constraints: none

Relaxable constraints: none

Prerequisite: none

Replannable?: nil

Know-one: To get to know whether or not some agent knows some concept, ask that agent yes or no.

Effect: (KNOW ?F ?S)

Operation: (YES-NO ?I ?G ?F ?S)

Nonrelaxable constraints:
(DESCRIBABLE? '?S),
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQ ?F ?G)),
(EQUAL ?F *PARTNER*),
(SET ?I (NEW-FRAME-ID))

Relaxable constraint:
(UNCERTAIN? (CNODE-STATE
(CNODE-WITH-NAME '?S (AGENT-CONCEPTS ?G))))

Prerequisites: none

Replannable?: t

Know-two: To get the self to know a concept, get the self to know some description of the concept.

Effect: (KNOW ?F ?S)

Operation: (KNOW ?F (DESCRIBES ?S ?D))

Nonrelaxable constraints:

(DESCRIBABLE? '?S),
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?G *PARTNER*),
(SET ?K (NEW-FRAME-ID))

Relaxable constraints: none

Prerequisite:

Applies-test:

```
(LAMBDA (PARAMS)
   (AND (LOW-RISK? (CP-CONTEXT-ARTIC PARAMS))
        (EQUAL ?G *PARTNER*)))
```

Content: (OPEN ?K ?F ?G ?S)

Replannable?: t

Know-three: To get the partner to know a concept, get the partner to know some description of the concept.

Effect: (KNOW ?F ?S)

Operation: (KNOW ?F (DESCRIBES ?S ?D))

Nonrelaxable constraints:

(DESCRIBABLE? '?S),
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?F *PARTNER*),
(BELIEVES ?G (DESCRIBES ?S ?D)),
(SET ?J (NEW-FRAME-ID))

Relaxable constraint:

```
(UNKNOWN-TO-PARTNER?
   (CNODE-STATE
        (CNODE-WITH-NAME '?S (AGENT-CONCEPTS ?G))))
```

Prerequisite:

Applies-test:

```
(LAMBDA (PARAMS)
   (AND (LOW-RISK? (CP-CONTEXT-ARTIC PARAMS))
        (EQUAL ?F *PARTNER*)))
```

Content: (OPEN ?J ?G ?F ?S)

Replannable?: t

Know-description-one: To get an agent to know some description of a concept, get some agent to tell the agent a description of the concept.

Effect: (KNOW ?F (DESCRIBES ?O ?D))

Operation: (INFORM ?I ?G ?F (DESCRIBES ?O ?D))

Nonrelaxable constraints:
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?F *PARTNER*),
(SET ?I (NEW-FRAME-ID))

Relaxable constraints: none

Prerequisite:

Applies-test: (LAMBDA (PARAMS) (LOW-RISK? (CP-DIFFERENCE PARAMS)))
Content: (FORALL ?N IN (OBJECTS-IN-PROP '?D) (KNOW ?F ?N))

Replannable?: t

Know-description-two: To get to know some description of a concept, get the agent to ask another agent to describe the concept.

Effect: (KNOW ?F (DESCRIBES ?O ?D))

Operation: (WH-ASK ?I ?F ?G (DESCRIBES ?O ?D))

Nonrelaxable constraints:
(TYPEP '?D 'PCVAR),
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?G *PARTNER*),
(SET ?I (NEW-FRAME-ID))

Relaxable constraints: none

Prerequisite:

Applies-test: universal
Content:

```
(FORALL ?N IN
   (COND ((BELIEFS-MATCHING (AGENT-CONCEPTS *SELF*)
                            '(DESCRIBES ?O ?D))
          (OBJECTS-IN-PROP (CNODE-NAME
                            (BEST-DESCRIPTION *SELF*
                               (BELIEFS-MATCHING
                                  (AGENT-CONCEPTS *SELF*)
                                  '(DESCRIBES ?O ?D))))))
         ((T NIL)))
   (KNOW ?F ?N))
```

Replannable?: t

Nested-know-one: To get the partner to know that the self has a concept, close the conversation about the concept.

Effect: (KNOW ?G (KNOW ?F ?P))

Operation: (CLOSE ?I ?F ?G ?P)

Nonrelaxable constraints:
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?G *PARTNER*),
(SET ?I (NEW-FRAME-ID)))

Relaxable constraints: none

Prerequisites: none

Replannable?: t

Nested-know-two: To get to know that the partner has a concept, close the conversation about the concept.

Effect: (KNOW ?G (KNOW ?F ?P))

Operation: (CLOSE ?I ?G ?F ?P)

Nonrelaxable constraints:
(TYPE-OF ?F AGENT),
(TYPE-OF ?G AGENT),
(NOT (EQUAL ?G ?F)),
(EQUAL ?F *PARTNER*),
(SET ?I (NEW-FRAME-ID)))

Relaxable constraints: none

Prerequisites: none

Replannable?: t

5.8 How the Plan Operators are Used

The domain plan operators given in the last section are primarily used the JAM system for planning, although, as we shall see in chapter 7, they are also used for replanning, repair, and goal adoption. In this section we explain how the planning works in order to demonstrate what constitutes a plan and how one is built.

Suppose that an agent, mary, has her currency pointer resting on a node which contains the unplanned goal (KNOW JOHN SECTION-ONE). The first thing mary does in her planning is look through her list of plan operators for those in which the effect unifies with her goal, and keep track of the necessary bindings. In this case, there are three possibilities; **know-one**, **know-two**, and **know-three**, all three with the bindings ((?F john) (?S section-one)). Next mary will work through the nonrelaxable constraints of each plan operator in turn, building up the bindings and possibly eliminating some of the operators from consideration.

We will only work through the constraints of the **know-three** operator here; the other two operators work similarly. First, the constraint (DESCRIBABLE? '?S) is checked by instantiating it to (DESCRIBABLE? 'SECTION-ONE). This is true, because objects are describable if and only if the planning language allows them in "describes" forms such as (DESCRIBES SECTION-ONE (LEFT-OF PALM-BEACH-0)); that is, if they are a map object (an object whose internal name ends with a digit) or one of the four sections of the route. Next, the constraint (TYPE-OF ?F AGENT) gets instantiated to (TYPE-OF JOHN AGENT), which is true. Then the constraint (TYPE-OF ?G AGENT) splits the existing set of bindings into two, since ?G can be instantiated two ways: ((?G john) (?F john) (?S section-one)), and ((?G mary) (?F john) (?S section-one)). The next constraint, (NOT (EQUAL ?G ?F)), immediately knocks out the former set of bindings. Then the constraint (EQUAL ?F *PARTNER*) is true for the one remaining set of bindings because mary is planning and ?F is set to john. The next constraint, (BELIEVES ?G (DESCRIBES ?S ?D)), extends the one remaining set of bindings for each description which mary believes to hold about SECTION-ONE, giving three possible binding sets:

1. ((?G mary) (?F john) (?S section-one) (?D (left-of palm-beach-0)))
2. ((?G mary) (?F john) (?S section-one) (?D (between palm-beach-0 swamp-0)))
3. ((?G mary) (?F john) (?S section-one) (?D (around swamp-0)))

Finally, the constraint (SET ?J (NEW-FRAME-ID)) adds to each possible set of bindings a value for ?J which is a new reference number for a game which might be constructed in the operator's prerequisite.

At this point, mary has found three sets of bindings which satisfy all of the operators nonrelaxable constraints; next she tries the relaxable ones. In this case there is only one, (UNKNOWN-TO-PARTNER? (CNODE-STATE (CNODE-WITH-NAME '?S (AGENT-CONCEPTS ?G)))), which is satisfied because mary believes that john does not already know the concept SECTION-ONE.[6]

Running the constraints of the other two plan operators eliminates them from consideration, and thus mary has three possible plans for her goal of (KNOW JOHN SECTION-ONE), one corresponding to each set of bindings. If at this point mary had no possible plans, she would reinvestigate each of the plan operators, this time ignoring the relaxable constraints. Since mary has more than one possible plan, she uses her task planning communicative posture settings to choose one of them. This mechanism is described in section 5.12.1.

Once mary has chosen a particular set of bindings to use, she will construct a list of goals and games which must be added to her plan. She does this first by examining the operator's

[6] Admittedly, the code for this arbitrary lisp constraint is opaque and we require a better interface to information about lack of belief than is afforded by the current arrangement.

prerequisites in the way explained in section 5.6.2. Suppose that mary chooses the set of bindings ((?G mary) (?F john) (?S section-one) (?D (between palm-beach-0 swamp-0)) (?J t4)). Assuming that mary has a low risk context articulation setting, she will construct one prerequisite, the game (OPEN T4 MARY JOHN SECTION-ONE), which, when executed, will cause her to make the move O_I(SECTION-ONE). Then she will construct the operator's operation by instantiating it to be (KNOW JOHN (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))). This means that her plan for the goal has two parts, the OPEN game and the knowledge goal. Mary can adopt this plan by pushing these two parts onto her plan tree as children of the goal (KNOW JOHN SECTION-ONE).

5.9 How Current Plan Costs are Maintained

The Principle of Parsimony which governs an agent's actions entails that agents always try to use the least cost plan which will lead to a satisfactory solution. This requires agents to be able to estimate the relative costs of their plans and the various alternatives. This section explains how agents keep track of the cost of the current plan as steps are added, changed, and executed. These costs are then used by the agent as part of the information they need in order to decide which alterations to a plan will complete the task most cost effectively. An explanation of how agents evaluate possible ways of continuing from the current point in a plan is deferred to section 7.3.

The cost of each node in an agent's plan tree is recorded on that node and updated as the plan is changed. The cost of a node which has been successfully completed or a game, whether or not it is complete, is zero. The cost of an unplanned goal is one (unless the same goal appears elsewhere on the tree with a higher cost, in which case the cost of the unplanned goal is that higher cost). The cost of an open goal (e.g., one that is planned but not yet fulfilled) is one plus the costs of its children plus the costs of any prior goals on the plan tree for knowledge of map objects which are mentioned by the open goal but not mentioned by its children. This last cost helps to make sure that the agent doesn't unintentionally underestimate the cost of a goal by "forgetting" that it has had trouble establishing an object mentioned by the goal at some earlier stage in its planning. The cost of an abandoned goal is not defined, since the costs only measure the current plan plus possible alternatives.[7]

[7] If the agent were to be able to consider reinstating abandoned goals, then the costing routines would need to keep track of their costs. This does not seem difficult but has been relegated to further work.

5.10 The Top Level Control Structure

The JAM system uses a very simple top level control structure which is borrowed from Power's thesis work [Pow74]. Control rests initially with a chair program whose function is to set up the initial conditions for the dialogue and then call the agents alternately. The top level call must name two agents, one with the route on its map and one without, and must designate one of those agents to "start the dialogue". The system does not allow the agents to have completely freely mixed initiative dialogues; such dialogues make modelling the discourse and planning more complicated because agents must try to match their own plans to their partners and eliminate any parts that overlap. This is an interesting research problem in its own right but is beyond the scope of this thesis since it is unnecessary for a simulation of the map task. For the most part, in the human dialogues one of the agents controls the course of the dialogue while the other provides responses, interrupting only when he or she can initiate recovery from a failure. Correspondingly, in JAM dialogues, one agent always has the top level goal of getting either itself or the partner to know how to get around the route, and its partner only ever initiates games in order to begin a recovery.

The chair program uses the following algorithm:

1. Initialise the agents involved in the current dialogue by reading in the maps using the method described in section 5.2.
2. Send whichever agent is to "start the dialogue" a message containing the goal (KNOW ROUTE-FOLLOWER ROUTE). Either agent can be sent this message, which instructs it to initialise its plan in such a way that it can begin the dialogue.
3. "Wake up" the agent who is to "start the dialogue".
4. "Wake up" the other agent.
5. Go to step 3.

Of course, this process does not terminate; modelling closing dialogues is left to further work. The closing dialogues in the current corpus are not very natural because of the way in which the data is collected and the fact that participants are not told what to do when they have finished the task.

Although this model is a very artificial one, it is not as limited as it may first appear. For instance, if we allow for the flexible mapping between intention and surface level behaviour that was discussed in section 5.4, the behaviour produced by this model fits the main features of the turn-taking process described by Sacks, Schegloff and Jefferson [SSJ78] and reviewed in section 2.2.2. That is, if we allow "ok" responses in informing games and "ok" feedback

in wh-question games to be realised at surface level by nods, certain kinds of eye contact, or nothing at all, then these moves can be considered in the turn-taking model to simply indicate transition-relevance places (TRPs) at which the partner did not make an utterance and the original speaker continued its turn using rule 1(c), which allows it to continue as long as no other agent self-selects. This means that one turn may span several "awakenings", as long as the partner has no substantial contribution to make to the conversation. If we were to expand the control structure to allow three or more agents, then the order in which the agents were awakened would create an unfair disadvantage that is not introduced in the turn-taking rules for all but the first agent awakened at each TRP. In this case, the first agent awakened would always be the "first starter" if it wished to speak, followed by the second agent awakened, and so on. However, this is not a problem for two-agent dialogue, since rule 1(b) takes precedence over rule 1(c); that is, the partner *always* has the right to interrupt and take control at every TRP. Thus for two-agent dialogue, as long as we make the assumption about the flexible mapping between dialogue games and surface level behaviour, even this simple control structure fits the turn-taking model.[8] This entails that for our purposes the fact that the chair program does not allow the agents to "think" simultaneously is not a major liability.

5.11 The Control for an Agent

In the last section, we explained how a chair program alternately "wakes up" one agent and then the other during the course of a dialogue. This section gives a very high level algorithm for what each agent does when it is awakened. We will return to this algorithm in sections 6.4.3 and 7.4 in order to fill in the missing details.

Steps 1 to 5 The agent reads and interprets the messages in its mailbox. At the start of the dialogue, the agent who is to start the dialogue will find one message in its mailbox which contains the goal (KNOW ROUTE-FOLLOWER ROUTE); it interprets this message by running an initialisation procedure that gives the agent that as a top level goal. Otherwise, the messages in the mailbox (if any) will be ordinary communications from the partner in the language given in section 5.4, and the agent will interpret them by adding the associated dialogue games (if the messages are initiating moves) to the discourse model or modifying games in progress (if the messages are responses or feedback moves) and updating its beliefs in line with having received the messages.

[8] Whether or not one agrees with the turn-taking model is another matter. The existence of channel conflicts seems to suggest that agents may not always recognise the partner's right to take over the turn or at least that they may not always notice a "first starter". Our choice of control structure is the main reason that we can not simulate channel failure.

Steps 6 to 9 The agent decides what action(s) to take during its turn, sends the appropriate message(s) to the partner, and updates its beliefs in line with having sent the messages.

Of course, the mechanisms by which an agent updates its beliefs and decides which actions to take are two of the major contributions of this research; the former is described in chapter 6 and the latter in chapter 7.

5.12 How Communicative Posture is Implemented

This section explains how each of the communicative posture parameters is implemented in the JAM system. The current version of the system only implements a representative sample of the parameters given in chapter 3: from task planning, the ontology and partner modelling parameters; from discourse planning, difference; from utterance realisation, context articulation; and the meta-planning parameter, plan commitment.

5.12.1 Implementing the Task Planning Parameters

There are two task planning parameters, ontology and partner modelling, which are implemented in the JAM system. The human versions of these parameters are described in section 3.2.1. In the JAM system, each agent uses these parameters to narrow down the set of possible plans and replans to the one of each which it believes to be most suitable. Then the agent will look at the best options put forward by each of the ways of continuing from the current situation (planning, replanning, repair, goal adoption, and communicating) and decide upon one of them based on cost. We have already described how the agent estimates the cost of the current plan in section 5.9; the way in which the agent estimates the costs of each of the possible continuations is given in section 7.3.

To understand how the ontology parameter operates it is important to remember that the Principle of Parsimony requires an agent's actions not just to be as efficient as possible, but also to suffice for the agent's current purposes. Because the map task leaves open the question of how accurately the route follower must draw the route, different human agents will choose different levels of detail for their explanations. These differences are not simply cost based, since some agents believe that less detail is necessary than other agents do, depending on how they have interpreted the task instructions. Our implementation of the ontology parameter models this difference in personality; low risk agents prefer more detailed explanations than do high risk agents. However, it is likely that estimated costs should affect an agent's choice of

amount of detail, especially since an agent may believe that it is more cost efficient to give a non-detailed description and have the partner request whatever elaboration is necessary than to give a detailed description in the first instance. This aspect of ontological choice is not modelled in our implementation.

The partner modelling parameter controls how closely the agent heeds its model of the partner when it builds an explanation; high risk agents do not mind whether or not all of the map objects are known to the partner as long as they are not definitely not known, whereas low risk agents distinguish between knowing that the partner has an object and not having any information about the partner's knowledge. In effect, this makes a high risk agent act as if the partner's map is exactly like its own except where it has information to the contrary. As with ontology, the implementation considers this to be a matter of how the agent has interpreted the task instructions. However, also as with ontology, it is likely that costs should at least partially determine whether or not the agent heeds a model of the partner, since it takes more effort to use a model than to ignore it.

In the implementation, agents use the two task planning parameters to narrow down the sets of possible plans and replans to one of each. The two task planning parameters thus determine what propositions will be used when agents inform their partners about some part of the task. They are also used when the agent chooses an appropriate answer to a wh-ask question so that the answer will be chosen in exactly the same way that the agent would have used if it had decided to inform the partner of the proposition spontaneously.

The planning and replanning strategies rate possible plans and replans in the following way. First all plans with the effect of the current goal (or replans with the effect of any of the open goals) are collected.[9] Then the plans are assigned ratings depending on what is believed about the map objects mentioned in them (for partner modelling) and the number of goals and games mentioned in the list of operations of the plan (for ontology). The best plan is the one with the highest rating. The rating for any one plan is the sum of its rating due to ontology and its rating due to partner modelling:

The ontology rating provides a measure of the complexity of the plan; low risk ontology settings prefer more complex plans and high risk settings prefer less complex plans. In low risk ontology, the plan is rated

5 for each object plus

1 for each goal or game mentioned in the operations list.

[9]The plan matcher uses unification to determine which plans have the required effect. Two plans unify if all of their subparts unify using the same substitutions.

In high risk ontology, the plan is rated

−5 for each object plus

−1 for each goal or game mentioned in the operations list.

These numbers ensure that the number of objects is always the dominant factor, since there are never more than four goals and games in the operations list. The low risk ratings prefer the most detailed plan. The high risk numbers are the inverses of the low risk ratings so that the least detailed plan is rated with the highest value.

The partner modelling rating provides a measure of how well tailored the plan is to the partner; low risk partner modelling settings prefer plans which are carefully tailored to the partner, while high risk settings do not spend effort determining which plan is most suitable. In low risk partner modelling, the plan is rated

.8 for each object mentioned in the operations list which is thought to be known to the partner plus

.7 for each object about which the agent is uncertain plus

.6 for each object which is thought to be unknown to the partner.[10]

Objects are considered to be mentioned in the operations list if they form part of a description in a goal or in any proposition (including responses and feedback moves) in a game in the operations list. The relative sizes of the numbers ensure that low risk agents prefer plans which refer to objects which are known to the partner, or at least those which are not known to be unknown to the partner. In addition, the ratings are set so that the contribution of the partner model to the overall task plan rating outweighs the contribution of the number of goals and games from the ontology parameter in cases where there are more objects than goals and games involved in the task plan. Within a single parameter, the relative ratings are designed to reflect what we think the most important factors should be; across the parameters, the relative ratings are designed to allow each of the parameters to be dominant under different circumstances.

In high risk partner modeling, the agent adds

.8 for each object mentioned in the operations list which is either thought to be known to the partner or about which the agent is uncertain plus

.6 for each object which is thought to be unknown to the partner.

[10] Here, knowledge is defined with "vividness" and lack of knowledge as "awareness"; these terms will be defined in section 6.1.

This setting has the same effect as the low risk setting, except that the high risk setting does not differentiate between cases where the agent knows the partner has an object and those where the agent is uncertain about the partner's knowledge.

As an example, suppose that a route giver has the current goal of (KNOW JOHN SECTION-ONE) and that the agent knows three descriptions for the first section: (BETWEEN PALM-BEACH-0 SWAMP-0), (LEFT-OF PALM-BEACH-0), and (AROUND SWAMP-0). Suppose that the agent is uncertain about the partner's knowledge of any of the map objects, as the agent would be at the beginning of a dialogue. Furthermore, suppose that the only plan operator which the effect that the partner knows how to get around the first section of the route is one which has one operation, the goal of getting the partner to know one particular description of the first section of the route.[11] Then the agent will have three possible plans for the given goal, each with one operator, and if the agent has low risk ontology and low risk partner modelling settings then the agent will rate these plans as follows:

The plan with the operation (KNOW JOHN (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))):

Ontology 5 for each object mentioned (10) plus 1 for each goal or game (1) makes 11.

Modelling .8 for each object which the agent thinks is known to the partner (0) plus .7 for each object for which the agent is uncertain about the partner's knowledge (1.4) plus .6 for each object which the agent thinks is unknown to the partner (0) makes 1.4.

Total 12.4.

The plan with the operation (KNOW JOHN (DESCRIBES SECTION-ONE (LEFT-OF PALM-BEACH-0))):

Ontology 5 for each object mentioned (5) plus 1 for each goal or game (1) makes 6.

Modelling .8 for each object which the agent thinks is known to the partner (0) plus .7 for each object for which the agent is uncertain about the partner's knowledge (.7) plus .6 for each object which the agent thinks is unknown to the partner (0) makes .7.

Total 6.7.

The plan with the operation (KNOW JOHN (DESCRIBES SECTION-ONE (AROUND SWAMP-0))):

[11] The formal version of this plan operator is given in section 5.7 with the name **know-two**.

Ontology 5 for each object mentioned (5) plus 1 for each goal or game (1) makes 6.

Modelling .8 for each object which the agent thinks is known to the partner (0) plus .7 for each object for which the agent is uncertain about the partner's knowledge (.7) plus .6 for each object which the agent thinks is unknown to the partner (0) makes .7.

Total 6.7.

Thus an agent with low risk ontology and partner modelling settings would choose the first plan, which has the most complex description. An agent with low risk ontology and either partner modelling setting would choose arbitrarily between the second and third plans by selecting the first plan with the rating 6.7 that it had evaluated. If the agent were to know that the partner does not have the swamp on its map, then the agent would prefer the second plan, since it does not use the swamp.

5.12.2 Implementing the Difference Parameter

In section 3.2.2.1, we explained that the difference parameter controls whether or not an agent assumes that any map objects which have not yet been discussed are mutually known. This affects whether or not the agent attempts to build an accurate model of the partner before it refers to such objects in the dialogue; low risk agents will try to acquire certain knowledge about those parts of the partner's map which are relevant to the current plan, whereas high risk agents will not. A very low setting in the human dialogues had the agents establish certain knowledge about every object on the map before beginning the main part of the task; this behaviour is not implemented in the JAM system. Instead, JAM simulates the behaviour of agents who, once they have chosen a possible description of the route or of the location of a map object, check to make sure that the partner will understand references to the objects involved before uttering the description. This is done by including the following prerequisite on the **know-description-one** plan operator:

Applies-test: (LAMBDA (PARAMS) (LOW-RISK? (CP-DIFFERENCE PARAMS)))
Content: (FORALL ?N IN (OBJECTS-IN-PROP '?D) (KNOW ?F ?N))

This prerequisite requires that agents with low risk difference settings know that whoever is hearing the description understands the objects which are used in it. If such an agent is planning to tell the partner something, then if the agent isn't already sure that the partner knows the objects involved, the prerequisite goals will force the agent to ask the partner yes-no questions such as "do you have the palm beach?". If the partner gives a negative response, then the agent

can either describe the location of the object or abandon its current plan. The applies-test of the prerequisite ensures that only agents with low risk difference settings try to establish the prerequisite knowledge; this allows high risk agents to "assume" that the knowledge is shared and risk a plan failure. As we shall see in section 7.2.4, in such cases the prerequisite can be used by the hearer in order to determine what repair is necessary in order to recover from the failure.

In the JAM system the difference parameter only controls references to map objects; it is not possible for agents not to know what descriptors such as "go up" or "to the left" mean because of the simplistic map representation described in section 5.1. Given a more sophisticated representation of the domain, the difference parameter could be extended to control first references to all new concepts introduced in the dialogue, whatever they might be. In this extension, one might want to use some measure of each concept's complexity in order to determine exactly how risky not having certain information about the partner's knowledge of the concept would be and to have a number of different difference settings which take risks at different levels of complexity.

5.12.3 Implementing the Context Articulation Parameter

In section 3.2.3.2, we explained that the context articulation parameter controls whether or not an agent explicitly opens dialogues when it changes the current context from one part of the task to another. In the JAM system, the context articulation parameter is implemented similarly to the difference parameter, in that both involve prerequisites on domain plan operators which are only activated when the agent has a low risk setting. The context articulation parameter affects the **know-two** and **know-three** plan operators, since these are the ones by which an agent begins planning for a new part of the task (a description of either a new section of the route or a new map object). The prerequisites require agents with low risk context articulation settings to initiate an OPEN dialogue game about the concept under discussion. This has the effect of adding utterances to the dialogue which simulate the surface level behaviour of the human dialogues, but it does not make much difference to the simulation itself. Since utterances in the agents' communication language always contain a referring expression for the concept under discussion, they can not be ambiguous in the way that a low risk context articulation setting is meant to prevent. Moreover, JAM agents can't use the OPEN game to refuse to talk about a particular concept and offer up a better plan, since they automatically return O_{RP} ("ok") whenever they hear an O_I move. Although this implementation does not satisfactorily reflect what happens in the human dialogues, we include it in order to show that better versions could still be implemented using the technique of selective prerequisites.

5.12.4 Implementing the Plan Commitment Parameter

The plan commitment parameter introduced in chapter 3 controls how readily an agent decides to replan, compared to deciding to continue the current plan (if no failure has arisen) or engage one one of the other recovery strategies (if there has been a failure). We have hypothesised that in the human dialogues, agents follow the Principle of Parsimony by choosing the way of continuing from the current situation which leads to the lowest cost or effort solution to the entire task. Thus the decision of whether or not to replan can be affected by making the costs of replans relatively lower or higher than the costs of the other options; agents with low risk plan commitment settings will want to penalise replanning heavily so that they tend not to replan, while agents with high risk settings will not. In addition, agents with low risk settings will wish to replan at the lower levels of the plan rather than higher ones, since this changes the plan the most conservatively, while agents with high risk settings will prefer to change the high levels of the plan. In the JAM system, the estimated cost of a replan depends on the agent's plan commitment setting and gives these results. More information about exactly how the costs are estimated are given in section 7.3, after we discuss exactly how replanning is implemented.

5.13 Summary

The implementations which we have chosen for the aspects of the JAM system described in this chapter restrict the resulting simulation of the human dialogues in a number of ways. First of all, the overly simplistic map representation and the agents' lack of spatial reasoning abilities entails that we will be unable to implement some communicative posture parameters (such as ontological resolution) and some recovery strategies (such as elaboration). The fact that the language which the agents use for communication is unambiguous with regard to what concept is being described entails that we will only be able to implement a limited version of the context articulation parameter and will not be able to implement context resolution at all. The style of interaction, which alternates between the agents via a chair program, means that agents can never have channel collisions. Finally, the requirement that only one agent may have the initiative in the dialogue at a time means that agents can not abandon each other's dialogue games or refuse to talk about particular concepts even if they believe it would be more efficient to do so. However, the implementation described in this chapter suffices for the purposes of the current work, and any limitations imposed by these aspects of it could be improved with further work.

Chapter 6

The Belief Model

Chapter 5 described a number of components of our agents which are necessary to support our dialogue mechanism, but which, for the most part, are too simplistic or are not novel enough to merit much consideration. Their descriptions have been provided primarily so that it is clear exactly how the system works and what problems it does not solve. We could have included a simple belief modelling mechanism among these components; however, the way in which our agents model beliefs is interesting in its own right, besides fitting in with the theme of operating according to the Principle of Parsimony. This chapter gives a general method for representing beliefs in dialogue systems and then describes the exact representation used by our agents.

Traditional belief systems (e.g., [Kon86], [Lev84], [FH88]) use theorem provers on a set of propositions and axioms in a logic of belief in order to determine how an agent's beliefs change as it learns new information. We believe that this approach is implausible as a model of how humans keep track of changes in belief. More specifically, we believe that the theorem proving approach provides a mechanism which is more general than that used by people when they work out the results of saying and hearing utterances on their beliefs. Although we can not make any psychological claims about what agents actually do when they calculate belief updates, the aim of our model is to provide a simulation which at least makes use of an intuitive organisation of the agents' knowledge in order to restrict their inferences to a plausible set. Theorem proving techniques do not take into account that humans can usually exploit both the form of the dialogue and the structure of the domain about which they are conversing to provide semi-automatic heuristic updating of their beliefs. There are two basic disadvantages of the theorem proving approaches:

1. In normal processing, the theorem proving techniques are less efficient than they could

be. Belief updates must be made by reference to the entire set of axioms and propositions in the logic of belief, without a partitioning of the relevant aspects as is suggested in this model. In addition, although methods have been developed for representing the effects of "mutual belief" without computing an infinite number of belief nestings (see especially Clark and Marshall [CM81]), the belief models for which these methods operate still do not capture the automatic nature of most simple human belief modelling.

2. Theorem proving techniques do not exhibit the same characteristics as human belief modelling when they are pushed through harder and harder computations. Intuitively, humans get "stuck" at some point in the computation and presumably must switch over to a less automatic mechanism, requiring processing overhead to change between the two techniques. Theorem proving techniques do not exhibit this switchover point, since they provide a uniform mechanism for the computations throughout.

Our model overcomes these disadvantages by exploiting both expectations about the dialogue and the structure of the domain in a heuristic mechanism which models the most common aspects of belief updating. Then the model can switch to the traditional approach, which computes the belief updates more precisely, when the heuristic approach fails, requiring extra effort both for the switch between the approaches and for the more complete updating. As a result of the bipartite structure, the model preserves the Principle of Parsimony by allowing the agent to choose a lower cost updating method which suffices for the current purpose. The heuristic approach can be considered to be a compiled version of a theorem prover and increases efficiency by decreasing expressive power.

Most of this chapter describes the heuristic belief mechanism, in which the agents use predictions about the immediate results of an agent uttering or hearing a conversational move to provide simple belief updating. The analysis makes use of the dialogue games described in chapter 5. It also makes use of an idea from intelligent tutoring systems called "degrees of acquaintance", which allows agents to make finer distinctions about the kinds of beliefs an agent can have about a concept than that between simply believing or not believing a proposition. Section 6.1 explains what we mean by "degrees of acquaintance" for the purposes of the belief model. Section 6.2 describes the way in which the belief model interfaces with the rest of the system. Section 6.3 gives an abstract description of the model. Section 6.4 gives a simple example of a belief representation which fits our model. Section 6.5 explains several different extensions to the heuristic belief model which allow agents to model uncertainty about what other agents believe. Section 6.6 discusses how the heuristic approach can be combined with the more complete theorem proving approach. Finally, section 6.7 describes the belief representation which is implemented in the JAM system. There are three appendices associated

with this chapter; appendix B examines the space requirements for our belief model, appendix C gives details about the exact belief model used by the JAM system, and appendix D explains how it is implemented.

6.1 Degrees of Acquaintance

Logics of belief traditionally operate in terms of "knowing" or "believing" propositions, the meanings of which have been discussed at length elsewhere (see Halpern and Moses [HM85] and Hadley [Had89] for two reviews). Traditionally the difference between believing something and knowing it has been that, by definition, no one can know something which is false, while one can believe anything at all. Since our agents, like all others, can never be sure whether or not their information is correct, we will not distinguish between the two terms and mean by them what is usually meant by "belief". In task-oriented dialogue, as in ITS systems, it may be useful to represent to what degree an agent "understands" some domain concept (i.e., some predicate, object, or proposition). How an agent brings up a concept, and even with whom it chooses to converse, will depend on how much knowledge the agent thinks that other agents have about the concept. There are a number of distinctions which can be made about the degree of an agent's knowledge, and we will call the different points "degrees of acquaintance".

The first easily distinguished degree of acquaintance is unawareness. One may well wish to distinguish concepts of which one's partner is totally unaware (as, for instance, I must be unaware of some subfields of molecular biology, although I can not name them, and I'm sure that my father is unaware of Gödel's Theorem) since this may affect how one introduces the concept to the partner. Unawareness doesn't really come into the map domain, but it could be useful in tutoring domains where the agents take the roles of experts and novices. There may even be degrees of acquaintance below unawareness, such as being mentally incapable of ever being made aware of the concept, but such properties aren't useful in the map domain. Many degrees of acquaintance above awareness are possible, and which ones are useful varies. Near the bottom of the scale, one might, for instance, remember from reading Steel and Reichgelt [SR90] that the term "brailing" has something to do with curtain ropes at a theatre, but not know anything more about "brailing". Then this knowledge isn't really useful to anyone except to place the term in a particular context. A higher degree of acquaintance might allow an agent to understand some references to an object but not all of them. For example, one might know that the Morning Star is Venus, but not that the Evening Star also is. At the top end of the scale, one might be able to decide yes or no for whether any description picked out the object or any instantiation were true for the predicate, and so be a total expert on that piece

of the world. There are many different possible degrees of acquaintance which could become important in different situations.

For the purposes of modelling communication among agents with goals, however, the most important degrees to distinguish are between agents who know something about the task and ones who know how to complete the task, since then one can evaluate the advice which the agents give differently. Thus when one has a burst pipe one might place more weight on what the plumber says than on what is said by the neighbours. It might also be useful to distinguish between an agent who knows "enough" about the topic for current purposes and one who knows too much, so that one can avoid getting too much advice! The condition which is relevant for both of these degrees of acquaintance is that of understanding a concept well enough to use it in executing plans which make use of the concept. Thus, what is important is not just how much information is known, but whether or not the right information is known. We shall follow Steel and Reichgelt [SR90] and call this condition "vividness". We do not provide a formal definition of vividness for humans here, but we assume that all agents must know what it means for a concept to be vivid to them, since otherwise they would not recognise when they understood some plan well enough to execute it. For some domains such as the map task, what it means for a concept to be "vivid" depends on the particular agent's notion of success. We also will not rule out that agents may consider some concept vivid for themselves and then change their minds based on subsequent information; that is, we do not assume that any "knowledge" is incontrovertible. For most planning domains, degrees of acquaintance which make use of the concept of "vividness" will play a central role in the agents' actions.

In this section, we have outlined a number of different possible distinctions that one can make about an agent's understanding of a domain concept. Which of these are useful depends on the particular domain and task involved. We could construct many degrees of acquaintance for any reasonably complex concept, but for most purposes it's probably only useful to have a rough stereotype of the other agents' knowledge. For instance, for reasons given in section 6.7, the JAM system will make use of only two degrees of acquaintance: awareness and vividness. Additionally, since the dialogue agents cannot do even elementary spatial reasoning, we can make no distinction between when an agent knows something about, for instance, the location of an object and when it knows everything about it. Cleverer agents or agents for other domains could well use finer distinctions, and their belief representations should be built accordingly. Our heuristic belief representation allows the designer to choose whatever granularity along the scale of acquaintance is necessary. The system designer must make this choice before the representation is implemented. This constraint is no more restrictive than the requirement in logical representations for the axioms involved to code the degress of acquaintance explicitly

before the system is run.

6.2 The Interface with the System

Before we can describe the belief model itself, we must explain what the model makes available to a dialogue system so that it is clear what purpose the model serves and for which other systems our model would be appropriate. The interface which is implemented in the JAM system is not as general and elegant as that which is allowed by the belief model in the abstract. Section 6.2.1 discusses what the abstract model allows and section 6.2.2 describes what is actually implemented. It would not be difficult to provide an interface with the capabilities discussed in section 6.2.1, but this is left to further work.

6.2.1 The Interface in the Abstract

There are two ways in which our abstract model is intended to communicate with a dialogue system:

1. The belief model provides a function which, given a move in the dialogue by either agent and a representation of the agent's beliefs before hearing or uttering that move, returns a representation of the agent's beliefs as they would be after the move. This function keeps the agent's beliefs in line with what has transpired in the conversation; agents could also use it to "predict" what effect an utterance will have when they are planning what to say. In the JAM system, agents call this function as soon as they "speak" (send another agent a message) or "hear something" (read a message from their mailbox), although this is not required by the belief model. This model does require belief updating to be done only as a result of some utterance and using one invocation of the updating function. This implies that the agent always determines all of the changes in its beliefs in "one go" and that these beliefs never change between utterances. Of course, this is an idealisation of the human case, but this is a general failing of belief models and not just of this particular model. Appelt [App85] (page 28) states that although human agents clearly are not logically omniscient, it is not unreasonable to model them as if they are since in practice they usually make all of the inferences which affect the current plan. We leave it to future work to determine a way to cut off the belief updating which is done immediately and resume what is necessary to planning as it is needed, but we believe that the way in which this model works makes such future work simpler than it would be using a theorem proving approach.

Of course, it is possible for systems to need to update their beliefs based on information from sources other than utterances. For instance, a system might want to take visual data into account. Although we will not have anything to say about such endeavours in this thesis, we do believe that it is possible to integrate such information in this belief model by formulating visual data as the world informing the agent, experiments as the agent questioning the world, and so forth.

2. The belief model allows the system to ask it what the agent believes. The dialogue system needs this information so that it can decide what action to take next, whether that involves planning a new game in the dialogue or responding to a game which is already in progress. The belief model allows any query about the state of a particular concept (such as "is the swamp vivid for me?" or "what degree of acquaintance do I have with the swamp?"), including those which involve nested beliefs (such as "do I believe that John has the concept of the swamp vivid?" or "what degree of acquaintance do I believe John has with the swamp?" or even "for whom do I believe the swamp is vivid?"). It also allows queries about all beliefs that match a given pattern (for instance, "give me all of the descriptions which I believe hold about the swamp" or "give me all of the descriptions which I believe that John believes hold about the swamp"). As one case of this, it allows yes/no queries similar to these but with all of the information filled in (such as "do I believe that the swamp is to the left of the palm beach?"). However, like most models, this model does not allow mutual belief to be checked by the most intuitive definition (i.e., that Mary believes X and believes that John believes X and believes that John believes Mary believes X and so on, ad finitum). All requests must always specify either the level of nesting being checked or an upper bound on the levels of nesting to be checked, or else this model (or any other belief model lacking the capability to reason meta-theoretically about its beliefs) would need to do an infinite amount of checking on successively higher and higher levels before returning the answer. Allowing requests which check over a number of nesting levels can be useful for implementing "heuristic" mutual belief checks which terminate after a set number of nesting levels.

Our belief model admits an interface which seems no more or less restrictive than those of other models, and so we would expect it to be useful for dialogue systems in general and not just for our own.

6.2.2 The Implemented Interface

The interface to the belief model which is implemented in the JAM system is less general than the one specified in the last section, primarily because the current belief model was developed after the rest of the JAM system had already been implemented. The first function of the interface, which returns a representation of an agent's beliefs after hearing or uttering a dialogue move, works as we have suggested in the last section. The second function, which allows the agent to ask itself queries about what it believes, provides a subset of the queries allowed by the abstract interface. It currently contains a set of routines which for any given concept returns a yes or no answer to the questions:

- Is the concept vivid for me?
- Do I think that the concept is vivid for the partner?
- Do I think that the concept is not vivid for the partner?
- Am I uncertain about whether or not the concept is vivid for the partner?
- Do I think that the partner thinks that the concept is vivid for me?
- Do I think that the partner thinks that the concept is not vivid for me?
- Do I think that the partner thinks that I am uncertain about whether or not the concept is vivid for the partner?

In addition, for the special case of knowing descriptions of concepts (for instance, knowing that "to the left of the palm beach" is a description of the swamp), the system provides a routine which will answer questions such as

- What descriptions do I believe hold about the X?
- What descriptions do I believe that the partner believes hold about the X?
- What descriptions do I believe that the partner believes I believe hold about the X?

and so on, plus yes/no questions about particular descriptions such as the following:

- Do I believe that the swamp is to the left of the palm beach?
- Do I believe that the partner believes the swamp is to the left of the palm beach?

These queries do not allow the system to generalise over agents (asking, for instance, "for whom do I believe the swamp is vivid?") or to abstract over a number of nesting levels at the same time. However, they suffice for the current purposes of the JAM system, and it would not be difficult to provide an implementation of the interface which is specified for the abstract model.

6.3 An Abstract Description of the Belief Model

Theorem proving methods for updating beliefs during the course of a dialogue use axioms about the effect of hearing a particular type of utterance. For the belief updates that we wish to model, two such axioms might be "if you are aware of X and some agent tells you a proposition about X and you have no reason not to believe the proposition, then believe it. Furthermore, if you understand the proposition and it contains "enough" information ("enough" varying depending on the situation), then mark X as vivid for yourself." In such a system, beliefs about how the agent stands in relation to X, how the agent thinks John stands in relation to X, and so on, are all represented separately. So are beliefs about how the agent thinks John stands in relation to X and how the agent thinks Mary stands in relation to X. Moreover, although no agent will believe that John is only aware of X and also that X is vivid for John, the agent has no way of using one piece of information to check both. Now, although a reasonable theorem prover will have some way of ordering the facts which it uses in order to constrain search, one natural organisation might provide *one* structural unit which contains all of the agent's information about the concept X. Rather than representing the truth or falsity of facts in a knowledge base, this belief model represents the "state" of the agent's beliefs concerning each of the agent's concepts.

Each agent uses state designations to mark everything it thinks about a particular concept. For instance, an agent might have concept X tagged with a state which means that X is vivid for the agent, the agent thinks X is vivid for Mary, and the agent thinks John is unaware of X. The number and meaning of the states depends on the number of agents involved in the dialogue, the number of degrees of acquaintance which the agent is to distinguish, and the number of nesting levels which the agent is to compute without falling back into a more complete model of belief. The human designer must determine ahead how many states the agent should represent, and therefore how much reasoning the belief model can do in the semi-automatic mode.

Once the meaning and designation of the states has been determined, the changes in belief that come about as the result of hearing or saying a dialogue move about a particular concept can be represented as state transitions. So, for instance, hearing the initiation of an informing game about concept X when X is in a particular state (say state one) will always cause X to

be in another state (say state three). This reduces belief updating for that particular concept to locating the concept in the agent's memory, reading off the state number, looking up the appropriate transition given that state and the dialogue move, and storing the new state. In addition, if the domain is represented as a network of related concepts, then belief updates for the entire domain can be computed by propagating them through the network. This mechanism can be considered to be a compiled version of a theorem proving model of belief.

After belief updates are computed in this manner, determining what the agent thinks about a particular concept involves reading the state of the concept and working through the agent's key to the state numberings to find out what that state means.

This mechanism assumes that the system designer knows ahead how many agents will be in the dialogue, how many degrees of acquaintance are needed for the domain, and how many levels of nested belief will be required. The first two of these are reasonable assumptions for HCI systems. Section 6.6 will discuss a way of weakening the third assumption by adding a method for dealing with nesting levels which are not represented within the state mechanism.

6.4 An Example Belief Representation

This section describes a belief model for an agent that wishes to distinguish three degrees of acquaintance (unaware, aware, and vivid) and to use two dialogue games (INFORM and WH-ASK, as defined in section 5.4. The state transition mechanism will keep track of beliefs of agent A in a dialogue with agent B up to one level of nesting (that is, representing both what A thinks and what A thinks B thinks about each concept).

6.4.1 Exploiting the Regularities of Dialogue

Given the requirements of the example, and stressing that no agent can represent a concept of which it is itself unaware, there are six possible states in which any concept C can stand:[1]

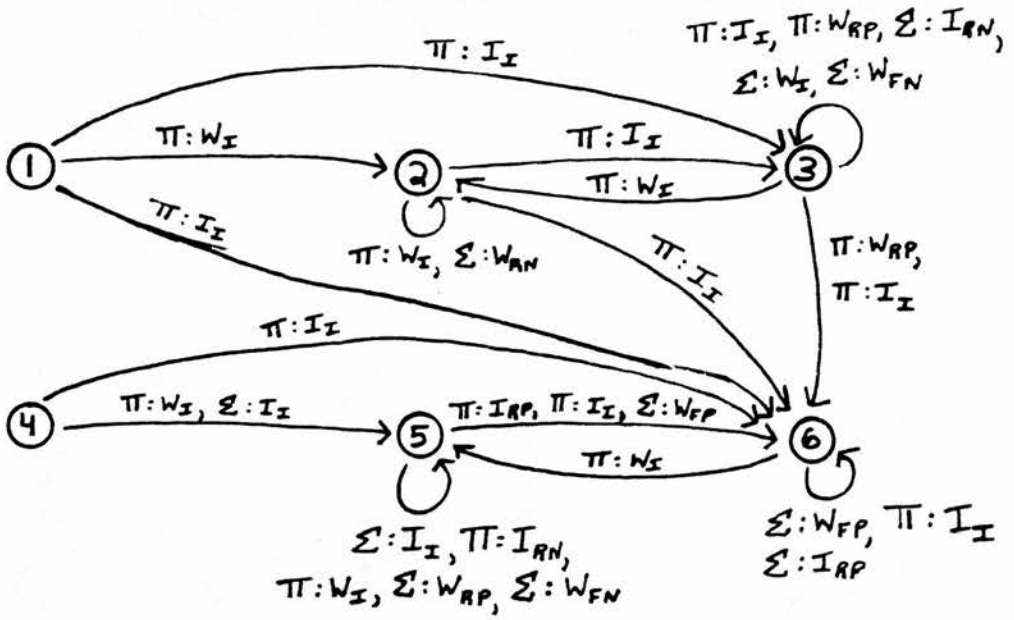
1. A is aware of C and thinks that B is unaware of C
2. A is aware of C and thinks that B is aware of C
3. A is aware of C and thinks that B has a vivid understanding of C
4. A has a vivid understanding of C and thinks that B is unaware of C

[1] We assume that an agent has a particular degree of acquaintance with respect to C if and only if it knows that it does, and so we do not include states such as "A is aware of C and thinks that A has a vivid understanding of C and thinks that B is aware of C". Most models of belief make a similar assumption. This is not necessary to our argument, but it does affect our space calculations for the model.

5. A has a vivid understanding of C and thinks that B is aware of C
6. A has a vivid understanding of C and thinks that B has a vivid understanding of C

Then the effects which agent A believes the allowable dialogue moves have on the belief state for that particular concept C can be expressed in the transition network shown in figure 6.1, by letting A be Σ (the self) and B be Π (the partner). For example, if agent A is in state four

Figure 6.1: A Six State Transition Network for Agent Σ



concerning concept C (i.e., C is vivid for A and A thinks that B is unaware of C) and agent B makes the dialogue move $I_I(C)$, then agent A will follow the arc labelled $\Pi : I_I$ from state four and end up with concept C in state six (i.e., C is vivid for A and A thinks that C is also vivid for B). The state network in effect "compiles out" the rules which agents would use to understand a dialogue into a very efficient form which works for a predetermined subset of the possible ways in which the rules apply.

The agent's choice of an arc to follow in the state transition network is entirely deterministic, although there are some cases in which there are two arcs with the same label from the same state in the network (e.g., the two arcs from state one labelled $\Pi : I_I$, one leading to state three and the other to state six). This only occurs on moves by the partner which introduce propositions about concepts, such as $\Pi : I_I$ and $\Pi : W_{RP}$, and the arcs always go to states three and six. In these cases, the agent can always decide which arc to take because the

difference between the two resulting states always depends on the agent's understanding of the proposition involved. If the agent "understands" the proposition (i.e., if the agent has vivid all of the concepts which combine to form the proposition), then it will take the arc to state six; otherwise, it will take the arc to state three. In general, if the agent understands the proposition, then it will move into a state which represents that it knows the concept which the proposition describes, and if does not understand the proposition, it will move into a state which represents that it does not know the relevant concept.

Another interesting feature of the state transition network is that it must allow the partner to initiate any game from any state in the network, even if there is no reason for the partner to initiate the game from that state. For instance, the network gives a transition from states in which the partner has a concept vivid for games in which the partner asks about that concept. We require these arcs just in case the agent is wrong about what the partner knows; without them, the agent would simply fail to understand dialogue about any concepts for which it started with the wrong beliefs about the partner's knowledge. Another way of implementing the same behaviour would have been to invoke a special procedure whenever the agent needs a missing state transition which corrects the agent's belief about the partner's belief in a plausible way and then searches the transition network again. In either case, these special cases could be marked in such a way that the number of times that the agent had changed its mind about the partner's belief about each concept could be counted; this information might be useful in deciding upon future actions, including in deciding to change one's communicative posture settings. We leave to future work further consideration of this extension.

In this representation, whenever a concept is in state six the agent will act as if that concept is mutually understood by the two agents unless further information indicates otherwise, either causing a change in the concept's state or making the agent reason to further levels of nesting using a technique such as that described in section 6.6. Our concern in using this representation is not simply to provide a representation that is tractable; Clark and Marshall [CM81] have shown that it is possible to handle the effects of "mutual belief" without infinitely many nesting levels. Instead, we are concerned about providing an implementation which is efficient for the inferences that agents typically make during dialogue. This is a benefit for HCI systems generally. Moreover, as we shall see in section 6.8, a side effect of the way in which we structure our model is that it more accurately reflects human processing than do previous belief models. Since mutual knowledge is the goal of most task-oriented dialogue, and since it is represented in this model by a state in a transition network in which agents make dialogue moves in order to traverse arcs, it should be possible to plan the dialogue using this same mechanism by reasoning backwards from (in this case) state six to the current state of the concept under

discussion. It is beyond the scope of this research to consider this possibility further, and so we use traditional A.I. planning techniques instead, as described in chapter 7.

6.4.2 Exploiting the Structure of the Domain

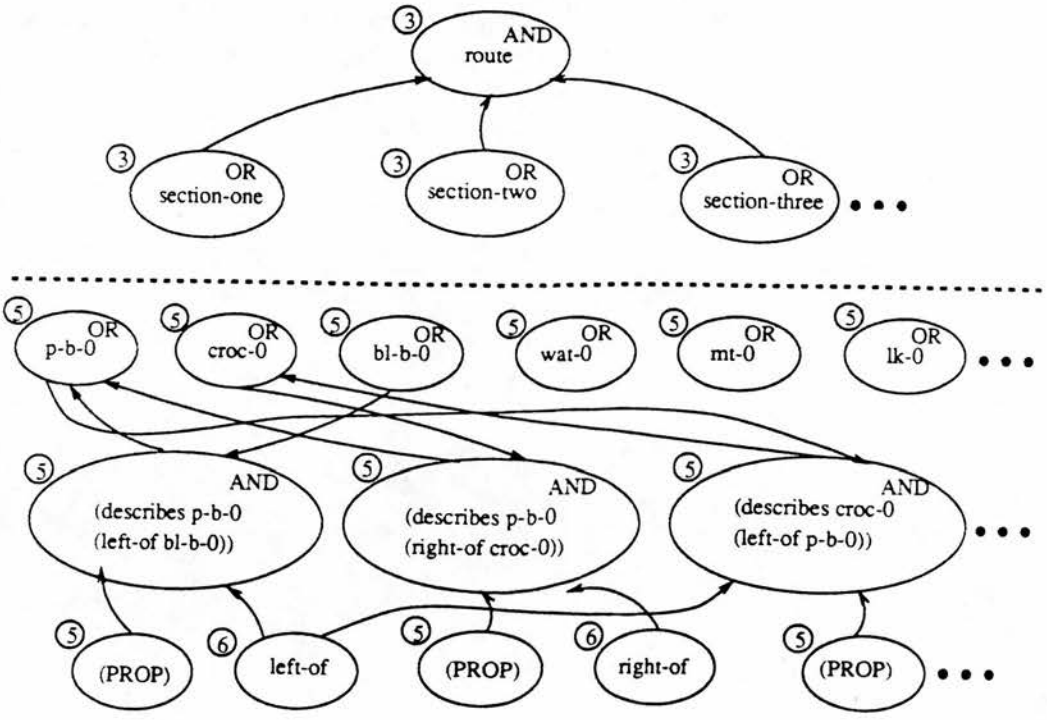
The mechanism we have described so far only gives the changes in belief regarding the concept which is central to the utterance. We still need some way of updating beliefs about concepts which are related to the central concept of the utterance. Work in user modelling has developed many common sense rules for reasoning about another agent's knowledge (for instance, see Kass [Kas88]) such as the following:

1. If the agent believes that another agent has vivid knowledge of P, then the agent believes that for some way of dividing P, the other agent has vivid knowledge of all the subparts of P.
2. If the agent believes that another agent has vivid knowledge of all the subparts of P for some way of dividing P, then the agent believes that the other agent has vivid knowledge of P.

There are five such rules which are used by the JAM system; they are listed in section 6.7. So far, we have reduced the problem of representing changes in belief from a theorem proving application with many axioms to one with just a few combined with a state transition network. This reduction is made possible by exploiting the regularities of changes in belief that come abut during the course of a dialogue. The problem can be further reduced to following state transitions and propagating activation through a network if we also exploit the structure of the domain.

Consider the changes in belief that come about as the result of hearing a particular utterance. Primarily, the belief state of the concept which is central to that utterance changes. In addition, rules like the ones we have given may change the belief states of related concepts. However, coding these rules as axioms in a theorem prover which operate over the entire space of concepts in the domain entails that the search for secondary effects may be costly. This is because such a representation does not show clearly which concepts are related to each other. Many domains, such as the map task domain and expert system domains like that of mineral exploration encoded in PROSPECTOR [DGH79], have a very clear structure by which concepts can be placed in a network of AND and OR nodes. A fragment of a concept network representing the route follower's knowledge before the beginning of a map task dialogue with the same maps as are used by the computer agents in the JAM system is given in figure 6.2. In the figure,

Figure 6.2: Part of B's Initial Concept Network



arrows point from a concept to its parents, and the number attached to a concept represents its current belief state. Any nodes with names ending in digits are objects which occur on the agent's map. For simplicity, both agents consider the route to be split into a number of sections which must be described separately, as described in section 5.1. The nodes named ROUTE and SECTION-ONE to SECTION-FOUR are special map concepts which correspond to the entire route and each of the four sections. Nodes which have names beginning with the token DESCRIBES are ones which relate the object which is described to some description of the object; we will refer to these as *description nodes.* Propositional "glue" nodes, labelled "(PROP)", simply indicate that in order to know the parent, which is always a description node, in addition to knowing the other children, one must also know how these children fit together to make up the description. Note that, since B's concept network is divided into two halves, agent B does not yet know how the objects on the map relate to the route. The goal of the dialogue is, in effect, to establish how the two halves are connected.

The domain structure provides a more efficient implementation of the inference rules for belief updating. For instance, the two inference rules given so far simply mean that for any agent, an AND node is vivid if and only if all of the node's children are vivid. Although the effects of these rules might be quite costly to calculate if the structure of the domain were not used, the rules can be implemented neatly as propagation of discrete functions up and down an AND/OR tree. The technique extends easily to networks of AND and OR nodes as long as propagation along a path a network is only continued if a change has been made to the current node, cutting off the infinite traversal of cycles. The rules which the JAM system uses for propagating beliefs are given in section 6.7; details about the implementation of these rules is given in appendix D.

6.4.3 The Agent's Algorithm upon Awakening

In section 5.11, we gave the algorithm which each dialogue agent follows when it is awakened by the chair program of the JAM system, but we omitted details about how agents compute changes in their beliefs. The mechanism with these details included is implemented as the procedure sketched below:

1. The agent retrieves the first message from its mailbox. There are only two possible kinds of messages. One is the "start" message from the chair, which triggers an independent initialisation procedure that gives the agent a top level goal to fulfill and skips straight to step 6. The other is an ordinary communication from the partner in the language given in section 5.4.

2. The agent adds any new concepts and connections suggested by the propositional content of the message. If the central concept of the message is unknown to the agent (i.e., if the agent can't decode the given referring expression to any known concept), then the agent adds a new concept for it with a state that indicates that the concept is not vivid for the agent but it is for the partner.[2] Then if the message is one that gives propositional content (the initiation of an informing game or a positive response in a wh-question game), the content is analysed and added to the concept network. This is done by going through each of the referring expressions in the proposition in turn, finding the denoted concept or creating a new one with a state that indicates that the concept is not vivid for the agent but that it is for the partner. In addition, if the proposition is a novel one, a description node is created (with a state that indicates that the concept is not vivid for the agent but that it is for the partner) which has as its children all of the concepts of the proposition plus a propositional "glue" node whose role is to remember in what order the concepts slot together in order to form the proposition. The description node is necessary to keep all of the different descriptions of a concept in different branches of the tree, thus maintaining a strict AND/OR structure. Propositional "glue" nodes are always created as being vivid for both agents, since we assume that once an agent has heard a proposition it understands how the subconcepts of it fit together even if it does not understand those subconcepts themselves. Finally, a connection is added into the concept network to make the new description node a child of the message's main concept.

3. Next, the agent interprets the main force of the partner's move by computing the change in belief about the central concept.[3] It reads the state of the central concept and then looks in its state transition network for all arcs leaving that state which are labelled with partner's move. In most cases there will only be one, in which case the agent simply stores the state which that arc goes into as the new state of the central concept. However, in

[2] This state is intuitively incorrect in some cases; for instance, if the partner asks "Where is the lake?", then one would expect the partner not to know where the lake is. This case doesn't occur in the map task, since in order for the partner to ask about the lake it must at least be on the agent's map, and therefore the agent will already have a concept for it in the network. However, even in other domains this discrepancy is unimportant from the point of view of the agent's behaviour because it is corrected immediately in the next step. It is left to further work to determine a better way of inserting new concepts into the network.

[3] In general, the central concept is the concept about which the agent is trying to elicit or give information. In the JAM system, the central concept of any move which is part of an opening, closing, informing or yes/no game is that concept which is named by the entire content of the initiating move of the game. For instance, in a yes/no game such as "Do you have the swamp?", the central concept is the concept about which information is sought (the SWAMP). In an informing game, the central concept is the description about which the initiator is attempting to be informative. Wh-ask games are a special case, since only an incomplete concept is available for matching until a positive response is made, if at all. In the JAM system, the central concept of a wh-ask game starts out as the concept for which information is requested in the initiating move (which is named after the keyword "describes" in the move's content), and, if a positive response is given, changes to the entire description given in that positive response halfway through the game. It would be possible to avoid this situation if wh-ask initiating moves were reformulated as games which request the partner to initiate an informing game in the next move.

some cases (W_{RP} and I_I moves) there are two arcs. In this case, the difference between the two final states is in whether or not the concept will be vivid for the agent itself, and the agent can decide between them by determining whether or not it "understands" the propositional content presented in the message; that is, whether or not it had to add any new concepts which are not vivid for it. When the agent has determined the ending state, it stores it as the new state for the central concept. In addition, then the concept which represents the propositional "glue" for that description is marked as vivid for both agents.

4. Now the agent must propagate the change in belief from the central concept to related concepts. This is done by applying a set of rules of the form "if a concept has changed state to state X and [other conditions], then change the state of all {children, parents} of that concept to state Y". Details of the rules used by the JAM system are given in section 6.7; how they are implemented is described in appendix D. At this point, the agent will have determined all of the consequences of the partner's move.

5. If there are still messages in the agent's mailbox, then it returns to step 1. Otherwise it moves ahead to the next step.

6. At this point, the agent must determine its next move, possibly using queries to itself about what it believes. The way in which the agent plans its actions is described in chapter 7; at the end of this step, the agent may have planned one move and sent it to the partner's mailbox. If it has planned and executed a move, then it goes on to the next step; otherwise, it has nothing to say and returns control to the partner.

7. Next the agent must determine how the main force of having made its move affects its beliefs about what the partner believes and what the partner believes it believes. This is done in the same way as in step 3, although the move has been made by the agent itself.

8. The agent must determine how its beliefs about any concepts related to the central concept of its move change. This is done in the same way as in step 4.

9. Now the agent returns to step 6 to plan subsequent moves in the dialogue.

In the JAM system, the agents have a top level procedure which encodes this algorithm and which is called whenever they are awakened by the chair program. More information about how the state transition network and belief propagation are done is given in appendix D. The concept network for the agent's knowledge prior to the dialogue is built from a simple database, as explained in section 5.2, using roughly the same procedure as step 2 but with stored initial states for the concepts. The transition network is implemented entirely by hand, although

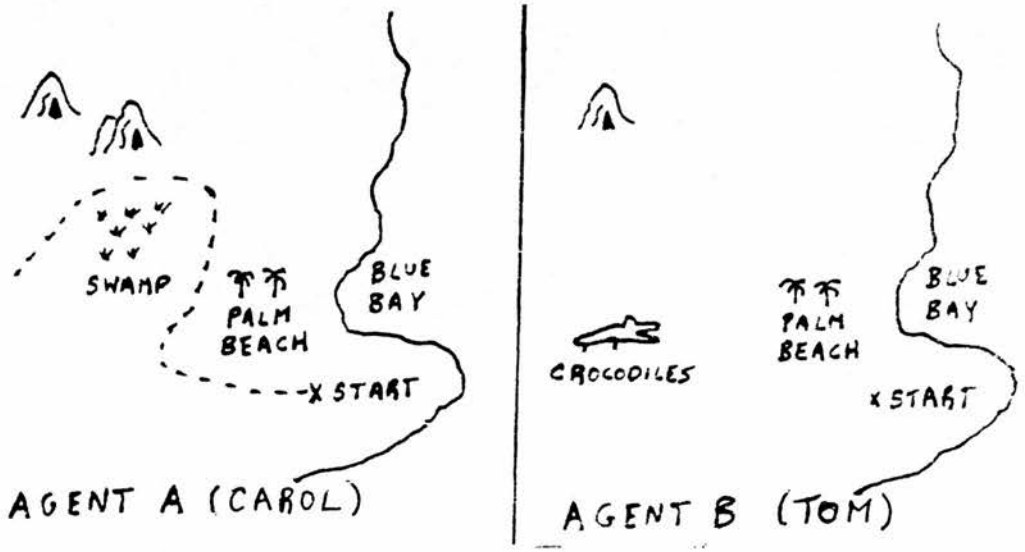
there is no reason why it should not be compiled from a logical representation; this is left to further work.

6.4.4 A Worked Example

This section gives a worked example in the map task domain of keeping track of belief changes using the six state transition network given in section 6.4.1. We will return to this example again in sections 7.5 and 8.3. Here, we analyse the belief changes from agent tom's perspective in the following dialogue extract, annotated with the English gloss provided by the JAM system:

(talk *carol* *tom*)	*(talk *carol* *tom*)*
carol makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*carol says to tom: the first section of the route goes between the palm beach and the swamp.*
tom makes an inform(r) move with content not-ok	*tom says to carol: i do not understand.*
tom makes a wh-ask(i) move with content (describes "swamp" ?var356)	*tom says to carol: where is the swamp?*
carol makes a wh-ask(r) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*carol says to tom: the swamp is between the waterfall and the palm beach.*
tom makes a wh-ask(f) move with content ok	*tom says to carol: ok.*

Figure 6.3: Parts of the Maps



The parts of both carol's and tom's maps which are relevant to this dialogue fragment are shown in figure 6.3.

When tom is first awakened by the chair program, he has the $A : I_I$ message in its mailbox. He retrieves that message and adds the new concepts and connections suggested by that message's propositional content. He already has the concept to which he can refer as the "first section" and which he knows as SECTION-ONE. He also already has the concepts "between" and "palm beach" (which it knows as BETWEEN and PALM-BEACH-0, respectively). However, he has no concept to which he can refer as the "swamp", so he creates a new one (which he gives some unique name such as T114) with a state indicating that the concept is not vivid for him but that it is for carol. Then he creates a description node (which he knows as (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 T114))) with a state that indicates that the concept is not vivid for him but that it is for carol. Furthermore, he makes the new description node have as its parent SECTION-ONE and as its children BETWEEN, PALM-BEACH-0, T114, and a newly created propositional "glue" node, with a state that records that it is vivid for both carol and tom, which indicates how the concepts get connected to build the description. This new description node is the central concept of the informing game, since it corresponds to what carol is trying to be informative about.

Next, tom computes his changes in belief from hearing the utterance. He first looks up the current state of the central concept of the move, (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 T114))), (state 3, which means that he does not have the concept vivid but thinks that carol does) and then looks for arcs from state 3 labelled with the move that carol made in the transition network ($\Pi : I_I$, which has Π because carol, the partner, made the move and I_I because that was the move given in the message). There are two arcs, one leading to state 3 and the other to state 6. In cases like this one, the agent must choose one of the two arcs depending on whether or not the agent "understands" the description of the concept (i.e., whether or not all of the concepts which combine to make up the description are vivid for the agent). Here, the description is not vivid because concept T114 is not vivid, and so tom follows the arc to state 3 and stores state 3 as the new state of the concept SECTION-ONE. In addition, since this was a move that introduced propositional content, he marks the propositional "glue" node as vivid for both carol and tom. In this case this behaviour is redundant because the node already has this state, but in cases where the node is not newly created (that is, where tom already knows the associated description) and the associated description has not been mentioned in the dialogue the state of the node may change.

Then tom considers propagating the changes in belief through the network beginning at the central concept under discussion, the description node. Since this node has become vivid for carol and it is an AND node, it marks all children of the node as vivid for carol. This change only affects the concept PALM-BEACH-0, changing its state to indicate that the concept is vivid

for both carol and tom. No other rules fire, and so the propagation ends.

Using an external mechanism, tom's dialogue planner then decides to close the game with the move $B : I_{RN}$ to indicate that the INFORM was not understood. To determine the consequences of his action, he follows the arc from the current state of the description node, (state 3) labelled $\Sigma : I_{RN}$ and discovers (since the arc also ends in state 3) that there are no changes to make, and thus also no propagation of the changes.[4] Then, again using an external mechanism, tom decides to initiate a new game with the move $B : W_I$("swamp" (since it wants to understand carol's original utterance) and executes that move. The central concept of the move is T114, since that is the concept about which tom seeks information. To assess the effect of the utterance, tom follows the arc labelled $\Sigma : W_I$ from state 3 (the state of T114), and discovers that the arc ends in state 3. Therefore he stores that the swamp concept is now in state 3, with no change. The external mechanism does not come up with any more dialogue moves to make, so tom returns control to the chair program.

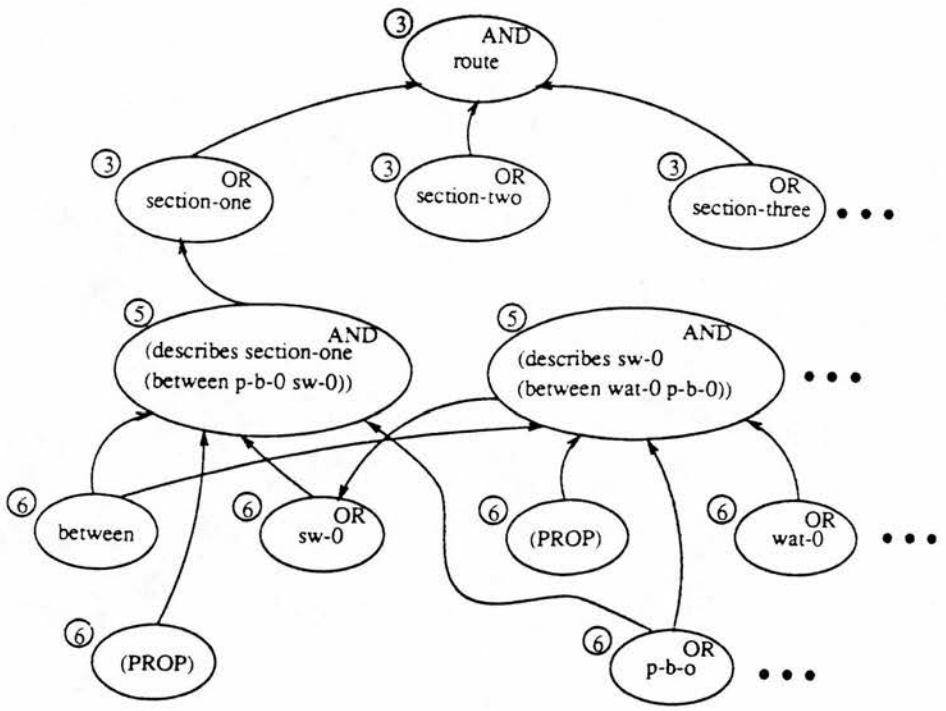
When tom is next awakened, his mailbox contains the answering move $A : W_{RP}$. He interprets this move using exactly the same technique he used to interpret carol's previous I_I move and changes the central concept of the game to be the new description node he knows as (DESCRIBES T114 (BETWEEN WATERFALL-0 PALM-BEACH-0)), since carol's reply has made this new information available. He then determines the changes in his own beliefs due to hearing the current utterance. He discovers that the new description node is in state 3 and so looks for the arc in the transition network which starts at state 3 and is labelled $\Pi : W_{RP}$. In this case, again there is a choice; there are arcs ending up in states 3 and 6, which tom can choose between depending on whether or not the concepts which combine to make up the given description of the swamp are vivid to him. In this case, since the necessary concepts (BETWEEN, WATERFALL-0, PALM-BEACH-0, and the propositional "glue" node) are vivid for tom, he follows the arc to state 6 and stores the new state in the T114 concept. This new state indicates both that the concept is vivid for tom and that tom thinks that the concept is vivid for carol. Because there has been a change in the state of a concept, tom must propagate the change through the concept network. He traces all of the pointers up out of the T114 node; in this case there is only one, leading to a node describing SECTION-ONE. Because this description node is an AND node, and because all pointers going out of the node lead to nodes which are vivid to tom, he marks the description node as vivid. Then he traces all the pointers out of the description node, in this case the one pointer leading to the node SECTION-ONE. Since the node SECTION-ONE is an OR node, and since there is one child of the node which is now vivid, tom marks SECTION-ONE as vivid to himself. Then he follows the pointers out of SECTION-ONE

[4] This does not imply that the move was uninformative; it only means that tom can not represent the changes in belief without resorting to further levels of nesting.

to find the route node, but since this node is an AND node and still has children which are not vivid, the propagation stops. None of the other propagation rules fire, and so the propagation ends.

After propagating the belief changes, tom decides according to the external mechanism to make the move $B : W_{FP}$. He follows the appropriate arc in the transition network to find no change to the belief states. At this point, as one would expect, the concepts SECTION-ONE and T114 are vivid for tom, and tom thinks that they are also vivid for carol. Part of tom's concept network after this section of the dialogue is given in figure 6.4.

Figure 6.4: Part of Tom's Concept Network After the Dialogue



6.5 Modelling Uncertainty

It is rarely the case in dialogue that agents already know what partners know about before they start. Therefore agents need to have ways of representing that they are not sure what degree of acquaintance another agent has with a particular concept. This section looks at several different ways that agents can represent their uncertainty about other agent's beliefs within the state transition network representation. There are two basic possibilities: one can simplify the situation by only considering what moves one would make from the relevant certain states, or one can represent uncertainty explicitly, by providing states which mean that the agent is uncertain. The former is the more efficient method, but it is not appropriate in all cases, and specifically, not for systems which will want to explicitly ask their partners questions about their beliefs to clear up any uncertainty before proceeding.

6.5.1 Explicitly Representing only Certain States

It is possible to deal with uncertainty by having the system pretend that it is in whichever certain state will provide the most beneficial behaviour. For instance, a generally low risk agent may decide that it is best to always act as if any object is not vivid for the partner unless the agent is sure that it is; in this case, the agent might use a single state to represent awareness and uncertainty. High risk agents might take the opposite approach, in effect behaving as if any concept is vivid for the partner unless it has explicit information to the contrary. Whichever choice is made, representing uncertainty by associating it with a particular certain state uses the least number of states, and therefore requires the least space. To be exact, let n be the number of nesting levels, a be the number of agents, and g be the number of degrees of acquaintance, and let S_C represent number of states needed if only certain states are represented, then[5]

$$S_C = g^{n+1}(a-1)^{\frac{1}{2}n(n+1)}.$$

If we assume that the typical two-agent HCI application might need to encode four degrees of acquaintance and three levels of nesting, we require 256 states. Any system which uses this kind of representation needs to have some way of choosing the most beneficial certain state to consider when uncertainty arises. However, this choice may not be as complicated as it seems at first. Participating in any dialogue about a concept is likely to bring the participants to certain knowledge about each other's belief states by the second move anyway, and so choosing a particular certain state in the first instance is not likely to change the course of the dialogue except in the short term. In systems with small state transition networks, the most beneficial

[5] Explanations of all equations in this chapter can be found in appendix B.

choice of a certain state might be the one that keeps open the highest number of moves. For systems with just two degrees of acquaintance, as in the JAM system, the best certain state to choose would be that in which one pretends that every other agent has the opposite awareness of the concept than oneself. That is, agents should assume if a concept is vivid for them then the other agents are only aware of it, and that if they are only aware of a concept then all the other agents have it vivid. This makes it possible for an agent to inform or ask other agents about the concept freely. In general, however, it will not always be possible to keep open all possible moves by choosing just one certain state from which to operate, since the agent might be uncertain between two states for which the set of allowable moves from one is not a subset of the allowable moves from the other. On the other hand, this restriction might be reasonable given that many agents have limited time or memory and will want to cut down the number of choices that they have to make about what to do next. After all, dialogue is exactly the kind of endeavour where risky behaviour is not just allowed but is favoured for efficiency reasons.

One of the major disadvantages of this approach from a representational point of view is that there is no way for the agent to indicate that it is not certain about its knowledge. Once it has chosen a particular certain state to use behaviourally, it "believes" that it is in that certain state and, as usual, allows for the fact that it may be wrong about the partner's knowledge. Agents can also handle uncertainty in a way that only requires them to explicitly represent certain states by always keeping and manipulating a set of the states among which they cannot distinguish with respect to the concept. This has the advantage that it is a truer representation of what the agent believes about the world, and allows the agent to answer correctly questions about its own beliefs. It also allows a simple implementation of the technique which we specifically could not get with the last approach; we can allow any of the moves which can be taken from any of the certain states in the set by considering the union of allowable moves from all the states in the set. This might lead to some unfruitful exchanges in the dialogue, but at least it will not block any useful ones. It seems reasonable that where there are not very many possible moves, agents might well employ this kind of technique. Furthermore, when allowing all possible moves is undesirable we can order these moves or eliminate some of them with the same type of technique needed for the last approach. These advantages are gained at the expense of the extra overhead needed to keep track of the sets and compute unions rather than keeping track of single states. However, dialogue about a particular concept tends to clear up uncertainty rather than create it, so in most dialogues this extra overhead should be acceptably small.

As we have mentioned, sometimes just being in an uncertain state causes an agent to make a move which is not allowed from any of the certain states among which the agent cannot

distinguish. For instance, exchanges like the following first mention of the graveyard are quite common in the human corpus:

MAP 10
G: have you got a graveyard?
R: yes

This game, which gives A certain information about B's knowledge, was only possible because A was uncertain in the first place. This exchange cannot arise if agents simply act as if they are in a particular certain state, and so the first approach we have described is severely limited. This game can be allowed if we use reasoning about a set of states by employing a procedure which adds it as a possibility whenever the set is not a singleton. However, at some point the procedure must do reasoning complex enough that it is worthwhile to compile out its structure to make the procedure itself efficient. In this case, the uncertain belief states should be represented explicitly as states in the transition network. The main advantage of avoiding using states to represent uncertainty is that it minimises the number of states and the complexity of the state transitions while probably still allowing successful completion of the dialogue fairly efficiently. We imagine that this approach might well be useful in a working system, although exploring it further is left to further work.

6.5.2 Explicitly Representing Uncertain States

The other main way of handling uncertainty is to create states which represent the different ways that an agent can be uncertain. That is, we might well have a state which by itself represents that, for instance, A is aware of C and isn't sure whether B is aware of C or has C vivid. Of course, this is potentially explosive if one considers every possible uncertain state. Suppose that we want to create a belief transition network for the JAM system which encodes uncertainty by having explicit uncertain states. Our requirements are to encode beliefs for two agents with two levels of nesting and the only acquaintance distinction being between awareness and vividness. Then, from A's perspective, there are eight certain states for any concept C:

1. A is aware of C, A thinks B is aware of C, A thinks B thinks A is aware of C
2. A is aware of C, A thinks B is aware of C, A thinks B thinks A has C vivid
3. A is aware of C, A thinks B has C vivid, A thinks B thinks A is aware of C
4. A is aware of C, A thinks B has C vivid, A thinks B thinks A has C vivid
5. A has C vivid, A thinks B is aware of C, A thinks B thinks A is aware of C
6. A has C vivid, A thinks B is aware of C, A thinks B thinks A has C vivid
7. A has C vivid, A thinks B has C vivid, A thinks B thinks A is aware of C
8. A has C vivid, A thinks B has C vivid, A thinks B thinks A has C vivid

Let us consider each nesting level separately. There are no possible uncertainties for zero levels of nesting; A always thinks it knows how A stands with respect to its own concepts. For one level of nesting, there is one possible uncertainty. A might not be sure whether B is aware of C or has C vivid. For two levels of nesting, the number of possible uncertainties increases because A could either be sure that B is uncertain about what A thinks, or A could be uncertain about what B thinks A thinks. There are seven possible positions that A could have regarding what B thinks A thinks of a concept C:

1. A thinks B thinks A is aware of C;
2. A thinks B thinks A has c vivid;
3. A thinks B is uncertain whether A is aware of C or has C vivid;
4. A can't decide between A thinks B is uncertain about what A thinks, A thinks B thinks A is aware of C, and A thinks B thinks A has C vivid (that is, A has no idea what to think about what B thinks A thinks);
5. A can't decide between A thinks B thinks A is aware of C and A thinks B thinks A has C vivid;
6. A can't decide between A thinks B thinks A is aware of C and A thinks B is uncertain about what A thinks;
7. A can't decide between A thinks B thinks A has C vivid and A thinks B is uncertain about what A thinks.

We can consider each of the levels of nesting as independent choices. Since there are two possibilities for zero levels of nesting, three for one level, and seven for two levels, there are forty-two possible states of belief for any concept. This number increases very rapidly with the number of nesting levels and a bit less quickly with the number of degrees of acquaintance. Using the same variable names as before, then if $S_U(n)$ represents the number of states needed if we are to explicitly represent all possible certain and uncertain states, $S_U(0) = S_C(0) = g$, since no agent can be uncertain about its own beliefs. Furthermore, for all $n > 0$,

$$S_U(n) = S_U(n-1)(a-1)^n U(n),$$

where $U(1) = 2^g - 1$ and for all $n > 1$,

$$U(n) = 2^{U(n-1)} - 1.$$

This gives us on the order of 10^{9870} states for our hypothetical HCI application ($g = 4$, $n = 3$, and $a = 2$), which clearly isn't practicable. We can cut down on the number of states if there are states between which we will never wish to distinguish. For example, if an agent always acts the same whether it is uncertain what another agent thinks it thinks or whether it thinks that the other agent is uncertain about what it thinks, then there is no reason to distinguish between these two states in the representation, and we can build a new state transition network which has one state representing the disjunction of these two positions. For example, in a two

agent system with two degrees of acquaintance and two levels of nesting, it might never be necessary to distinguish between the four different kinds of uncertainty which can arise at the second nesting level. In this case, we would only need eighteen states for the transition network instead of forty-two: aware or vivid for the zeroth level, and aware, vivid, or uncertain in an unspecified way for the first and second levels. The JAM system handles uncertainty using this technique. In general under this choice the space requirements are only slightly worse than those for only representing certain states. If we let S_E represent the number of states needed to represent all kinds of uncertainty at each level using only one state, then

$$S_E = g(g+1)^n(a-1)^{\frac{1}{2}n(n+1)}.$$

This gives us 500 states for our hypothetical HCI application ($g = 4$, $n = 3$, and $a = 2$). Of course, it is not always appropriate to conjoin different states. Which states can be conjoined in the network depends on the particular domain and the complexity of the dialogue required. However, in cases where the type of uncertainty is unimportant, it is possible that one of the approaches which does not explicitly represent uncertain states might be appropriate and more efficient. Representing the uncertain states explicitly allows the system designer to tailor the allowable moves to the domain and situation, with the disadvantage that each state combination must be considered case by case. Whether a representation using only certain states or one that represents uncertain states explicitly should be used depends on how individualised the moves for each uncertain state should be.

6.6 Combining the Network with a Theorem Prover

The belief state mechanism assumes that the system designer knows in advance how many agents will be in the dialogue, how many degrees of acquaintance are needed for the domain, and how many levels of nested belief will be required. The first two are reasonable assumptions; the last is not. One might need to reason to depths beyond those which are ordinarily used in conversation when, for instance, one needs to plan how to complete a task given a community of agents with different specialties and different information about who specialises in what. Even in the map task, agents might want to reason through several depths in order to recover from complicated failures which involve ambiguous referring expressions. It is necessary to interface the state mechanism to a complete theorem prover for belief exactly for those cases where we need to reason to a depth beyond that provided for in the state mechanism. We do not consider this to be a flaw of the state mechanism simply because the state mechanism is designed to be quick where people are quick. There comes a point in reasoning (e.g., when one works on logical puzzles over a certain complexity) where people have to work out their beliefs explicitly and

encounter cognitive difficulty; we propose that at this point, the simulation should switch over to the complete theorem prover. When we require the theorem prover, running the two systems together will be slower than running the theorem prover separately because of the overhead involved in switching between the two mechanisms. However, this mirrors human behaviour where an agent gets "stuck" in a computation which is more difficult than the agent expected; after an initial failed attempt, effort is expended simply in determining which solution methods are appropriate. Accordingly, we have no efficiency requirement for the interface except that the computation involved should be feasible. The remainder of this section describes a simple interface which suffices.

Let us assume that no matter which mechanism is in use, agents immediately determine all logical consequences of their beliefs up to the level which they have chosen to represent.[6] Furthermore, assume that beliefs at level n are unaffected by beliefs at any level higher than n.[7] Let n be the number of levels of belief represented by the state mechanism. Then an interface is possible between the state mechanism and a theorem prover which uses the state mechanism to compute all beliefs in levels n and below.

One simple approach runs as follows: As the dialogue progresses, the agent runs the state mechanism in the usual way, keeping a history of utterances in the dialogue. The first time that the agent needs to run a query at a level m greater than n (for instance, during a complicated recovery), the agent should run all the utterances through the theorem prover to level m, but using the state mechanism wherever possible and saving a list of all consequences at levels $n+1$ to m. The agent should also place a pointer to the dialogue history tagged with the value m at the last utterance in the dialogue. Then the next time the agent needs to run a query at a level p greater than n, it should look backwards through the dialogue history for the first pointer it finds with a tagged value of p or greater, run the theorem prover from that point up to level p, adding the new consequences to the old list of higher level beliefs, and place another pointer. If the entire dialogue has been processed up to level p already, then it suffices to look up the belief on the list.

---

[6] This is a common assumption in theorem proving approaches and is maintained in the network propagation techniques which we have given. We use the assumption here to constrain the interface so that it is only ever necessary to switch between the two mechanisms as a result of an utterance in the dialogue. However, it seems likely that interfaces can also be constructed for the general case.

[7] That is, whenever we want to calculate beliefs at a particular level, we never have to consider the beliefs at any higher level. A weaker version of this assumption runs as follows: Suppose that there is a function F which takes us from levels of belief to levels of belief which indicates the highest number of levels to which we must reason to make sure that we have covered all implications which change beliefs at the input level. Then there is always some cutoff level after which we know that no more processing is necessary. All theorem provers must make some version of this assumption in order to terminate, and so in real terms this isn't a restriction at all. However, it's clear that sometimes the strong version is too restrictive; consider the case where one realises that everyone else thinks that Labour is the best party to lead the country after the next general election, so that one comes to believe that too. It remains for further work to consider what changes must be made to the interface if we choose the weaker form of the assumption.

We have sketched this approach in order to convince the reader that such an interface is possible; improving upon it and implementing it is left to further work.

6.7 The Belief Representation for the JAM System

The JAM system requires a belief model which can keep track of dialogues between two agents with two degrees of acquaintance, awareness and vividness. Since in the map task agents become aware of concepts as soon as they are mentioned in the dialogue, unawareness would only be necessary if the agents were to build realistic first referring expressions (possibly using "a" to refer to objects of which the partner might be unaware, and "the" along with a unique description to refer to objects which are known to the partner). Furthermore, the agents' representation of the map is too coarse to make finer distinctions between awareness and vividness useful. The deepest nestings we expect our agents to encounter will be two levels, which, as we shall see in chapter 7, are used to handle the closing coordination after recovery from failure. It is not difficult to construct cases in which three or more levels of nesting would be useful,[8] but JAM's current plan operators do not require belief modelling beyond two levels. We have also chosen to represent uncertain knowledge imprecisely by allowing one state at each level of nesting to denote that the representing agent is somehow uncertain about what the partner thinks about the concept. We made this choice because we do not require different behaviours for the agents in the face of different kinds of uncertainty. Given these choices, the agents have an eighteen state representation; the state transition network is given in appendix C in tabular form.

Our agents use the following rules to propagate their beliefs through the concept network:

1. If some AND node C becomes vivid for some agent A (either the agent or the partner), then mark every parent node of C (which is an OR node) as also vivid for A.

2. If some OR node C becomes vivid for some agent A, then for every parent node P of C (which is an AND node), if all other children of P are also vivid for A, mark P as also vivid for A.

3. If some AND node C becomes vivid for the partner, then mark every child of C as also vivid for the partner. The equivalent rule for the self is not necessary because it should

[8] For instance, suppose that after a plan failure in which some agent B does not understand an utterance because B does not have some presupposed knowledge, the partner A makes a repair and then wishes to convey to B the fact that he or she believes that the recovery is complete. This requires A to model what A thinks B thinks A thinks B's degree of acquaintance with the relevant concept is.

never occur that a node becomes vivid for the self except by all of its children being vivid.

4. If some OR node C which was in an uncertain state becomes certainly not vivid for the partner, then mark every parent node of C as also not vivid for the partner. The equivalent rule for the self is not necessary because one can't be uncertain about one's own state.

5. If some OR node C which was in an uncertain state becomes certainly not vivid for the partner, then mark every child node of C as also not vivid for the partner. As with rule four, the equivalent rule for the self is not necessary because one can't be uncertain about one's own state.

6. If some AND node C changes so that the partner thinks that it is vivid for some agent A, then mark every parent node P of C (which is an OR node) so that the partner thinks P is also vivid for A.

7. If some OR node C changes so that the partner thinks that it is vivid for some agent A, then for every parent node P of C (which is an AND node), if the partner thinks that all other children of P are also vivid for A, then mark P so that the partner also thinks that P is vivid for A.

8. If some AND node C changes so that the partner thinks it is vivid for some agent A, then mark every child D of C so that the partner also thinks that D is vivid for A.

Here "marking" a concept to have a new property means changing the state of the concept to a new state which preserves the properties of the old state for all nesting levels that are not mentioned in the rule, but has the new property for the nesting level which is mentioned. Of course, other rules are possible,[9] and there is much scope for further work in this area. How these rules are implemented and called is described in appendix C. These rules are analogous to ones used by Kass [Kas88] and others in Intelligent Tutoring Systems, but our domain structure differs sufficiently from most tutoring domains that our rules are not directly equivalent to other existing sets. ITS domains tend to concentrate on the relationship between concepts and instances of concepts, whereas our domain does not have this level of abstraction. In addition, most tutoring domains do not have the same AND/OR structure which the map domain has, since in general for most concepts the system would not wish to assume that the student understands a concept if the student knows just one description for the concept. Our rules are also slightly more complicated than Kass' in two ways: they must handle our explicit representation of uncertainty, and they must allow not just for what the agent (system) thinks

[9] In particular, rules four and five could be generalised to allow any move to a certainly not vivid state.

the partner (student) knows, but also for what the agent thinks the partner thinks the agent knows. Our rules 2 and 7 are related to Kass' goal generalisation rule, and our rules 3 and 8 are related to his agent rule. Kass' primary agent, expert, and direct statement rules, which affect the agent's beliefs directly after hearing an utterance, are embodied in our belief transitions.

6.8 Conclusions

The model of belief used in the JAM system improves on previous models by

- taking advantage of the regularities of belief changes in dialogue in order to draw standard inferences quickly and automatically, and
- taking advantage of the structure of the domain in order to perform any inferences about belief changes by propagating them through related concepts, which greatly reduces the amount of search which is necessary.

In taking advantage of dialogue expectations and the domain structure, the belief model loses some of the flexibility associated with previous models. Most specifically, it must set an upper limit on the number of nesting levels for the beliefs which it can process. This would be a disadvantage for the model if it weren't for two facts: most dialogue can be handled with a very limited amount of nesting, and people can't process highly nested beliefs in the same automatic fashion as they handle ordinary conversation. We suggest that it is parsimonious for agents to use the kind of heuristic belief model presented in this chapter in most cases, falling back on a more complete model when the heuristic model fails. Just as with dialogue strategies, it can be better to take a fast, high risk approach to belief updating and risk having to expend extra effort in getting to the desired solution than to start out with the higher cost process in the first place. Our model provides an interesting comparison with other research developments in two ways:

- it gains efficiency in a way that is analogous to other AI systems, and
- it uses the same kind of two process model which is sometimes postulated for human processing.

Our model is more efficient than previous models because it is able to "compile out" the behaviour of inference rules expressed in a logic of belief which usually encode dialogue expectations and the domain structure. This work is similar in approach to systems in other domains

which increase efficiency by decreasing expressiveness in an initial deterministic mechanism and allowing processing to switch to a more complete mechanism if needed. One example of such a system is Marcus' PARSIFAL [Mar80], which uses a deterministic mechanism to parse all sentences "which people can parse without conscious difficulty" and which he claims could be hooked up to some higher level "conscious" grammatical problem solving component to handle garden path sentences. Another is Appelt's KAMP planner[App85], which uses "action summaries" to provide a STRIPS-like formalism to search the space of possible plans quickly but then drops through to a more complete possible worlds axiomatisation of the domain to make sure that the proposed plan will work. Thus our belief model is similar in approach to steps which people have taken to gain efficiency in very different kinds of AI systems.

Besides increasing efficiency, our belief model handles quickly inferences about beliefs which are not very deeply nested and and takes longer for ones that are, providing an interesting comparison with human belief updating. We hypothesise that human belief updating uses the same kind of two process model as this belief representation, and that this is true because it is more parsimonious for humans to do so. Such a hypothesis is not new; for instance, Clark [Cla89] suggests that there are two modes of operation for human thought – a quick, subconscious mode which can best be modelled using parallel distributed processing, and a slow, conscious mode which resembles models in a classical cognitivist framework. It also seems that human belief updating requires extra processing to handle unusually difficult tasks, and that there is some boundary point between simple, automatic calculations and harder problems, where people may get "stuck" for a while and incur overhead if they use the simpler approach where it is inappropriate. Our belief model is a step towards such a two mode process and maintains the Principle of Parsimony.

Of course, our model also has its drawbacks. The most important of these is that since our belief representations must be constructed by hand, building one is much more time-consuming than writing a theorem proving model of belief. In addition, once the belief representation has been constructed, it's difficult to see why it works. In section 9.5, we suggest that it would be both useful and interesting to build a compiler which would produce one of our belief representations from a theorem prover and some control information specifying the size of the representation to be produced. Unless work along these lines is successful, this kind of representation will continue to be hard to build and unmotivated. Also, despite the fact that our model is intended to capture the kinds of restrictions which naturally occur in human belief updating, it is still too powerful in an important way. Human agents often fail to make connections between a belief update for some concept and a belief update for some related concept, especially if that relationship is not very strong. Although the restrictions which have been proposed for theorem

proving belief models address this point, our model does not. We think that "decay" on the propagation of belief changes within the concept network is a more intuitive way of thinking about this kind of inference failure than the ones which have been proposed for theorem provers, but we have done no work in this area.

Chapter 7

Recovering from Plan Failure using a Layered Architecture

This chapter describes one approach to building an incremental planner for discourse which interleaves planning and execution and which can use recovery strategies, such as the ones described in chapter 4, when a failure occurs. Previous approaches to incorporating recovery strategies in AI planners have exhibited less flexible behaviour than that which is required for the human simulation; we have reviewed these approaches in section 2.4. Most work to date has concentrated on one recovery strategy at a time (e.g., Peachey and McCalla [PM86]; Wilkins [Wil88]), eliminating the need to determine which of a number of strategies to apply. Moore's work [Moo90] allows repair, reinstantiation, and replanning, but her system is not general enough to generate the behaviour observed in the human dialogues. Her system requires the strategies to be applied in a particular ordering until one of them succeeds; human agents are able to choose an appropriate recovery strategy more flexibly than working through a set list. In addition, Moore's replanning capability always tries to replan bottom level goals before top level ones, while human agents can choose any unsatisfied goal for replanning. Finally, the structure of her system requires that recovery strategies be accessible only when a plan failure is evident. Human agents occasionally replan even when no failure has occurred. These discrepancies occur because her system has a very rigid way of deciding what to do next in all situations. This chapter suggests that a more appropriate way to choose the next action is based on the estimated amount of effort that remains to complete the task using each of the different applicable options *next*. In addition, rather than having a number of different procedures for recovering from plan failures which can be called into action under set conditions, this chapter suggests that in order to simulate the flexible behaviour observed in the human corpus, these

procedures should be recast as possible ways of continuing from the current point in a plan and that they should all be considered at every choice point. The system could then use its cost estimates to always make the best choice of what to do whenever it is confronted with a choice at all, and could even choose to, for instance, replan if no failure has occurred but if the initial cost estimate for the current plan has turned out to be inaccurate.

This chapter goes on to describe the plan revision aspects of the JAM system, which are able to choose a recovery strategy more flexibly than Moore's system by evaluating possible continuations from the current point in the dialogue. Modelled on Stefik's MOLGEN planner [Ste81] but adapted for a domain that must interleave planning and execution, it integrates the recovery strategies as meta-planning operators which manipulate the lower level plans. Then a higher level operator can choose sets of recovery strategies to adopt at particular times, and yet another level can vary the way in which these sets are chosen. This system has the advantages that it is easy to change the way in which recovery strategies are employed and that repair and replanning, as well as the other recovery strategies and normal planning and communication, have the same status in the system.

7.1 Where Moore's Approach Fails

Johanna Moore describes in her thesis [Moo90] a planner for text generation which can repair, reinstantiate, and replan if necessary. Her planner reacts to the dialogue partner by allowing text plans which have failed to be corrected before the next piece of text is generated. Her system chooses a recovery strategy to apply according to a strict ordering. If any repair is possible then it is done. If not, then every reinstantiation is tried. Otherwise, replanning is used as a last resort, starting with the lowest level open goal and working upwards. This heuristic works well in domains where adopting a totally different plan for a goal is "expensive", either because the first steps of a plan involve the greatest part of the effort and repairs are likely to work, or because changes in plans can confuse the coordination among agents. However, this heuristic is too rigid for some planning situations. Consider a system for assembling a bookcase that knows about nails and wood screws. Suppose that the system attempts nailing two pieces together first, since that's easier than using wood screws, but that it accidentally bends the nail. By Moore's heuristic, the planner will continue using the nailing plan until there are no nails left. After a number of failed attempts, a more rational agent might conclude that for some reason unknown to it the nailing plan costs more than the effort of drilling the hole and putting in the wood screw. Certainly some time before the planner tries its hundredth nail, it ought to switch to the wood screw plan. This evaluation can be accentuated if the nail

holes threaten to ruin the wood. This example shows that a planning system which can both reinstantiate and replan must have flexible enough control that it can choose between the two options sensibly. A similar argument can be constructed which shows that there is no way to choose ahead of time to always try repair either before or after reinstantiation or replanning.

Moore's heuristic is too rigid to account for some of the recovery decisions which agents make in the human corpus. Consider the following initial segment of a dialogue:

1G: Right. Start from the sandy shore,
1F: OK.
2G: moving down... straight down.
2F: How far?
3G: Down as far as the bottom of the well.
3F: I don't have a well.
4G: Ah. Right, eh. Move down, eh, vertically down about a quarter of the way down the page. Move to the right in ... Do you have local residence?
4F: I do.
5G: Right, well, move up and round and above them.
...[with no further mention of the well]

In this dialogue, it would be possible for G to repair the plan by telling F where the well is. However, G chooses instead to replan and give F a totally different description of this section of the route, presumably because G thinks it will take less effort to complete the goal of getting around this section of the route by starting with the new plan involving the local residence than by repairing the old plan. Although there is not enough of the human data for any particular pair of agents to show whether or not the same agents make different choices in different situations, it seems unlikely that any one agent always uses the same ordering, and that G simply uses a different one than the one postulated by Moore.

Another problem with Moore's heuristic is that replanning and reinstantiation are always attempted at the most specific goal that has failed. In the woodworking domain, suppose that after reinstantiating the nail ten times, Moore's planner runs out of nails and decides to replan. It then replans the bottom level plan of attaching the two pieces of wood together and attempts the goal using wood screws. However, if the top level goal is to have a bookcase, under some circumstances it might "cost" less to abandon the lower level goal completely and buy one ready made. A more flexible replanning capability would be able to consider for what purpose the lower level goal is being attempted and consider changing the top level plan so that the goal is no longer necessary.

What Moore's system lacks is an estimate of the amount of effort needed to complete its goals using each of the various methods. This estimate should depend on the current situation; for instance, in the woodworking domain, someone who has tried nailing a bookcase together ten

times and failed will consider the plan of nailing the bookcase together differently from someone who is just beginning the task. Once an estimate is formed based on the current situation, it is most rational for the system to take the option which it believes requires the least effort to complete the task satisfactorily or which provides the best "value for effort".

However, it is not enough to simply extend Moore's system with a cost estimating function which will choose rationally among the different recovery strategies when a plan failure has occurred. Human agents sometimes replan even when no plan failure has occurred, and such a system would not be able to simulate this behaviour. A preliminary version of the JAM system extended Moore's work in such a way that it pushed repairs onto the plan whenever they were appropriate and that the system considered replanning at every planning step. This produces the desired cost-based surface level behaviour, but it had the effect that occasionally repairs were pushed and then immediately abandoned. In addition, since the preliminary version always chose a recovery strategy based exclusively on cost, it was not easy to investigate different decisions and to simulate other behaviours in the human corpus. What is required is a system in which the agent can choose flexibly among all of the different ways of continuing from the current situation. The architecture which we propose in this chapter and which is implemented in the current JAM system elevates the recovery strategies to have the same status in the system as other ways of continuing from the current point in the dialogue and then allows any possible continuation to be considered at any point, providing more flexible behaviour that more closely reflects the human corpus.

7.2 How the Strategies are Implemented

Chapter 4 gave a number of different recovery strategies which are employed in the human corpus; the JAM system will simulate three of these: replanning, repair, and adopting the partner's goals. These three recovery strategies, together with planning and communicating something to the partner, represent all of the different actions which a JAM agent can decide to take during the course of a dialogue. However, before we can explain how these strategies are combined in the overall system, we must explain how each of them is implemented. In this section, for each strategy we describe first how the system decides whether or not the strategy applies in the current situation and then what the system does if it decides to use the strategy; section 7.4 explains how an agent decides which of the applicable strategies to use. Four of the strategies, planning, replanning, repair, and adopting the partner's goals, refer to the domain plan operators, which axiomatise what effects domain actions are expected to have. These operators and the meaning of the language in which they are expressed is given in section 5.6.

Examples of each of these strategies in use are given in section 7.5.

7.2.1 Communicating

An agent can decide to communicate something to the partner if some other action which it has taken has made it plan to initiate a dialogue game or if there is currently a dialogue game in play in which the agent is to make the next move. An agent can recognise the first situation as that in which its currency pointer rests on a node containing a dialogue game which has status "uninitiated"; in this case communicating involves only retrieving the initiating move which has been placed in the dialogue game, making it, updating its beliefs in the way given in section 6.4.3, and marking the node as "initiated". An agent can recognise the second situation as one in which the currency pointer rests on a node containing a dialogue game which has status "initiated" or "replied".[1] In this case, the agent must construct a suitable move, make it, update its beliefs, and mark the node as either "executed" (if the move is the final move of a game) or "replied" (if the game still requires the partner to make a feedback move). Agents construct responding and feedback moves as follows:

I_R: If the agent "understands" the propositional content of the associated I_I move, then it responds with I_{RP}; otherwise it responds with I_{RN}.

W_R: If the agent has some belief which unifies with the associated W_I move, then it finds the belief which unifies with it which evaluates most highly according to its task planning communicative posture settings, as given in section 5.12.1 and returns it as the propositional content of a W_{RP} move. Otherwise, it responds with a W_{RN} move.

W_F: If the agent "understands" the propositional content of the associated W_R move, then it responds with W_{FP}; otherwise it responds with W_{FN}.

Y_R: If the agent has the concept given as the content of the associated Y_I move, then it responds with Y_{RP}; otherwise it responds with Y_{RN}.

O_R: The agent makes the move O_{RP}.

C_R: If the agent holds vivid the concept given as the content of the associated C_I move, then it responds with C_{RP}; otherwise it responds with C_{RN}.

[1] As we shall see in section 7.4.4, JAM agents always make moves in dialogue games as early as possible and immediately cede control to the partner if they make a move which requires a response. This ensures that any nodes with these statuses contain games in which it is the agent itself and not the partner which is to make the next move. This is not necessary for the system's operation, but it does simplify things somewhat. If we accept the argument about the mapping between dialogue games and surface level behaviour given in section 5.4, then providing an immediate response can be achieved by saying or doing nothing, and thus our requirement that agents make dialogue moves as early as possible is not a major limitation.

7.2.2 Planning

An agent can plan from the current situation if its currency pointer rests on a node with status "unplanned" (i.e., a goal which has not been planned yet). It plans by looking at all of the domain plan operators to see if any of them match the current situation; a plan operator matches the current situation if and only if its effect unifies with the current goal in such a way that all of the operator's nonrelaxable constraints are satisfied (as explained in section 5.6). Of course, there may be several plan operators which apply to any one goal, and even several instantiations of the same plan operator. In case of conflicts, agents always prefer plans in which all of the relaxable constraints are satisfied. In addition, if an agent has several such plans, it will choose the one which is evaluated most highly according to its task planning communicative posture settings, as described in section 5.12.1. Once an agent has chosen the best plan available and decided to use it, planning proceeds as follows: the agent pushes as subnodes of the current goal node any prerequisites of the plan which apply given the agent's communicative posture settings and the operations of the plan, and marks the current goal node as having the status "open". It also stores the name of the operator by which the plan was constructed in the node containing the goal for future reference. Then the agent will naturally move to the first unfinished child of that goal (i.e., the first prerequisite of the new plan) the next time the currency pointer is updated.

7.2.3 Replanning

An agent can consider replanning in the current situation if there is any node in its plan which has the status "open" (i.e., is a goal which has been planned and on which the agent is currently working, so that the goal has not yet been fulfilled, since in this case the agent might want to consider a new plan for the unfulfilled goal). The "open" nodes of a plan are exactly those nodes which are ancestors of the current point in the plan, since work on any nodes to the left has been completed and work on any nodes to the right has not yet been started. Note that these conditions do not require a plan failure to have occurred. At any one "open" node, the agent considers whether or not there are any replans which are better than the current plan by using almost the same technique as it used when it was planning originally. First it gathers together all instantiations of all domain plan operators which have the desired effect, but, since the agent has already planned for the goal at least once, it makes sure to eliminate any previous plans from consideration.[2] Then, in the same way as for initial planning, it chooses the best

[2] A possible replan is rejected as having been used previously if all of its operations are already children of the goal under consideration, regardless of ordering. The replan's prerequisites are not considered in this test. This test suffices for the current version of the JAM system, but it is conceivable that more subtle combinations

possibility out of those that remain.

Of course, it is perfectly possible for there to be more than one "open" node in the plan, and thus the agent needs a way to compare replans at different locations. JAM agents use the most straightforward method, which involves considering all of the open nodes from the top down.[3] This can be considered to be a simple instance of dependency directed backtracking [CRMM87] because of the way in which plans are structured. The only open nodes in the plan are the ancestors of the failed node (indicated by the currency pointer), and, since nodes always serve the purpose of furthering the goals of their parents, the set of open nodes is exactly the same as the set of nodes which depend on the failed node. Of course, chronological backtracking over the entire plan structure would not be sensible because any plan nodes which are to the left of the node indicated by the currency pointer have already been completed and should not be undone. Moore's replanning [Moo90] is equivalent to chronological backtracking over just that part of the tree which has not yet been completed.

There is one more complication in the way that JAM agents decide upon the best replan: domain plan operators contain a flag which, if set, marks any plans made using them not replannable. This flag enforces that, given a decision to satisfy a goal, the system accepts that the first plan which it adopts for the goal will always be the best way to satisfy it. Setting the flag on any particular node to nil does not force us to complete the associated goal; as long as some ancestor of the node was planned with an operator whose flag was not set, it is possible that the node will be abandoned when that ancestor is replanned. It is up to the designer of the domain plan operators to decide which operators should have the flag set and which should not. We have included this facility since it seems likely that there are some plan choices which human agents do not reconsider because it would cause too much cognitive effort to do so. Such a facility might also be useful for increasing efficiency in more traditional planning domains.

Once the system has determined the best replan to use and has decided to use it, replanning happens as follows: the agent marks as abandoned all of the unfinished descendants of the goal for which the agent is replanning, moves the currency pointer so that it rests on that goal, and then pushes the new plan onto the plan tree in exactly the same way as it would if it were an original plan. It also stores the name of the operator by which the plan was constructed in the replanned node for future reference. Then the agent will naturally move to the first unfinished

---

of domain operators would require more subtle testing.

[3] It might suffice, however, to stop looking once a replan is found which has a cost that is lower than the cost of the current plan. With a good cost estimating routine, this should ensure that the first replan which is found to be better than the current plan is also the best one, since the cost estimates are intended to reflect the best possible complete realisation of each plan. This extension is left to further work.

child of that goal (i.e., the first prerequisite of the new plan) the next time the currency pointer is updated.

7.2.4 Repair

An agent can consider repair in the current situation if the currency pointer rests on a node with the status "open". This entails that the plan must have already been completely executed but failed, since if the plan had not been attempted then the pointer would have been on some subnode of the goal node rather than at the goal itself, and if the plan had not failed, then the goal node's status would have been "succeeded", not "open". Repair involves filling in missing prerequisites to a plan after the main part of the plan has already been executed, and so repair only applies if a plan failure has already occurred; otherwise it would always be most cost effective to move on to the next part of the task. Unlike with replanning, JAM agents only consider repairs for the lowest level goal that has failed; repair does not involve abandoning any parts of the current plan, and so moving up levels in the plan tree will not leave a failure behind. JAM agents also only consider repairing the current plan for a goal and not previously abandoned plans. There is only one possible repair for each plan, and so unlike planning and replanning, the agent does not need any evaluation routine to decide which repair to make. To determine what the repair would involve, the agent retrieves the name of the plan operator which is stored at the current goal and lists all of the prerequisites of the operator which are goals[4] *as if its communicative posture settings were entirely low risk.* Then it eliminates all prerequisites on the list which were included in the plan before it was executed, since these prerequisites will already have been satisfied. Prerequisites are considered to have been included in the plan if they unify with a goal contained in one of the children of the current node.

Once an agent has determined what repair will be necessary and has decided to make the repair, it does so by pushing all of the missing prerequisites onto the plan tree as subnodes of the current goal node. Then the agent will naturally move to the first missing prerequisite the next time the currency pointer is updated. Note that this method of repair only suffices in domains like simple dialogue where it is possible to make the repair by executing prerequisites *after* execution of the main plan. In domains where prerequisites set up conditions which are necessary for the plan's operations to work at all and not just for them to be more felicitous, repair will be a more complex strategy and will perhaps involve re-executing the plan's operations.

[4] This restriction only ensures that agents do not revive the prerequisite of opening the current part of the dialogue by using an open game, as required by the low risk context articulation setting. It works because the only prerequisites which involve dialogue games in the current system are for opening games. Clearly this is not the correct restriction, but it suffices for the current purpose.

7.2.5 Goal Adoption

Our implementation of goal adoption is rather less intuitive than our implementations of the other recovery strategies. It uses a simplified version of Allen's plan recogniser [All83], which was reviewed in section 2.4.2. An agent can consider adopting one of the partner's goals in the current situation if the currency pointer rests on a node containing a dialogue game which the partner has just initiated or if the currency pointer rests on a goal node which the agent has obtained by goal adoption. An agent can recognise the latter condition because the goal adoption routine will have set a flag on the node when the goal was adopted. In either case, the agent checks to see if goal adoption is applicable by looking for a plan which, when considered from the partner's perspective, "explains" the partner's behaviour. It begins by enumerating all of the possible plans it has which have an operation that unifies with the partner's goal or dialogue game. Then it eliminates any of these plans which do not satisfy the operator's relaxable constraints *from the partner's "perspective"* (that is, when the agent substitutes the partner for itself and its beliefs about the partner's beliefs for its own beliefs in the plan which the agent itself would use to bring about the goal). At the end of this procedure, it will have only those plans which it thinks the partner could be using given the partner's behaviour. Given the JAM system's current domain plan operators, there is only ever one plan identified by this procedure; given a richer set of operators, the procedure would have to be extended to identify one plan out of a possible set of plans. Then the partner's goal is the effect of that plan. As a safeguard against pushing a goal as a subgoal of itself, the agent will only consider adopting the goal if that goal is not contained in an ancestor of the current node. The opposite situation, where the adopted goal contains itself as a descendant, can not occur given the plan recogniser just described and the current version of the planning operators.

Once the agent has made its best guess at the partner's goal and has decided to adopt that goal, it does so by inserting a new node containing that goal into the plan tree between the current node (which was either a dialogue game initiated by the partner or another goal adopted from the partner) and its parent. In addition, the agent sets a flag on the new node to indicate that the new goal has been adopted from the partner. The JAM version of goal adoption does not have the agent push any subsequent operations from the recognised plan onto the plan tree, so JAM agents can not take over control of one of their partner's plans. In particular, this rules out dialogues such as

> A: The first section of the route goes to the left of the palm beach.
> B: OK. Where is the second section?

because although in the JAM system B would recognise A's goal that B know how to get

around the route, B would not initiate any parts of a plan for that goal. Simulating dialogues such as this one requires some method for controlling agents in which both agents can take the initiative at the same time, which is beyond the scope of this research. However, JAM agents *can* initiate repairs for failed plans which were initiated by the partner; an example of such a dialogue is given in section 8.3.

Once a goal has been successfully adopted, the goal adoption routine works recursively from it in order to determine why the partner might have that goal. Thus one invocation of the goal adoption routine may add an entire chain of ancestor goals to the plan tree which explain a single utterance of the partner's. We have chosen to implement goal adoption in this way because it seems unlikely to us in such a simple domain as the map task that agents bother to reason about why their partners say what they do without taking such reasoning to its logical conclusion. However, it is a simple change to make goal adoption be controlled one step at a time.

7.3 Estimating the Cost of a Continuation

In order for an agent to choose which strategy to apply next in any one situation according to cost, the agent must first have a way of estimating the costs involved. JAM agents construct cost estimates for planning, replanning, and repair; this section explains how these cost estimates are made. JAM agents do not construct estimates for communication or goal adoption, and so the agents must use other means for determining whether or not to apply these strategies. The cost estimates for continuation strategies are only ordered plausibly with respect to each other and are not intended to reflect costs in absolute terms. They are only intended to provide relative costs for each of the possible continuations so that the least costly option can be chosen. Once they have been used to determine the agent's next action, they are discarded and the new planning steps which make up the continuation are added to the current plan using the way of estimating current costs which was given in section 5.9.

The continuation strategies always provide the estimated cost to complete the entire task given that the strategy is used next; this is to ensure that recovering at a low level in the plan is not automatically favoured over recovering at a high level simply because the recovery takes less effort by not finishing as much of the task. Thus the cost of each of the continuation strategies is constructed by reference to what changes the strategy makes to the cost of the current plan. The most desirable continuation is the one with the lowest estimated cost. Planning and replanning at any level use the same basic procedure for constructing an estimate of the cost of the new planning step which is added: they add up the number of operations in the plan that

are goals. The cost of planning is estimated as the old cost of the top level goal of the entire plan tree (in the JAM system, always the goal for some agent to know how to get around the route) plus 1 for each goal which must be added to the tree as part of the new planning step, plus 500 if the plan does not satisfy the relaxable constraints on the domain plan operator used. The cost of 1 for each goal favours plans which involve as few goals as possible. The 500 point penalty is intended to ensure that plans which satisfy the relaxable constraints are used whenever possible, since (given the current planning operators) 500 is greater than the cost for any unrelaxed plan will be. Similarly, the cost of a replan at any level is the old cost of the top level goal minus the cost of the node which is being replaced (i.e., the number of goals which would be abandoned in the replan being costed) plus 1 for each goal which must be added to the tree as part of the replacement planning step, plus 500 if the replan does not satisfy the relaxable constraints on the domain plan operator used, plus a threshold value for the "hidden cost" of changing plans. For agents with high risk plan commitment settings, this threshold is 1 plus 1.6 for each level of plan which must be abandoned, making high risk agents replan relatively easily and prefer to replan at the highest level possible, all other things being equal. For agents with low risk plan commitment settings, the threshold is 11 minus 1.6 for each level of plan which must be abandoned, making low risk agents replan seldom and prefer to replan at the lowest level possible.

The mechanism for estimating the cost of a repair is somewhat simpler. The repair is considered to have the old cost of the top level goal plus the number of missing prerequisites that would be pushed onto the plan plus 1 for each level from the top that the repair occurs. This penalty embodies the principle "if things are getting complicated, they'll only get worse", and keeps the agents from getting pushed into ever more complicated plan structures as they repair repairs and so on. As the number of levels in the plan increases the cost of the repair eventually overtakes the cost of a replan. This overtaking comes early for agents with high risk plan commitment, since 1 plus 1.6 for each level to be abandoned is relatively small compared to the number of plan levels at a typical failure, and late for agents with low risk plan commitment, since 11 minus 1.6 for each level to be abandoned is relatively large compared to the total number of levels. The exact numbers are designed to allow, among other things, the comparison between the two conversations given in section 8.6, and are useful only for this particular domain.

For an example of estimating the costs of possible continuations, suppose that two agents, doris and fred, have just had the following part of a dialogue:

Figure 7.1: Doris' Current Plan

```
open: (know fred route) <6>
      open: (know fred section-one)  <2>
         open: (know fred (describes section-one
                             (between palm-beach-0
                                      swamp-0))) <1>
            executed: (2198 inform
                            doris fred
                            (describes section-one
                               (between palm-beach-0
                                        swamp-0))
                            not-ok ?var688) <0>
      unplanned: (know fred section-two) <1>
      unplanned: (know fred section-three) <1>
      unplanned: (know fred section-four) <1>
```

doris makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*doris says to fred: the first section of the route goes between the palm beach and the swamp.*
fred makes an inform(r) move with content not-ok	*fred says to doris: i do not understand.*

Then, given the way of maintaining plan costs given in section 5.9, doris' current plan is represented by the tree in figure 7.1, with costs given between angle brackets:

Furthermore, doris' current position in her plan is at the goal (KNOW FRED (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))). Suppose that doris has a high risk plan commitment setting. At this point, there are four possible ways for doris to continue. She can repair the current plan by pushing onto the tree the two prerequisites which she skipped when she attempted the current goal ((KNOW FRED PALM-BEACH-0) and (KNOW FRED SWAMP-0)), or she can choose one of three possible replans for the goal (KNOW FRED SECTION-ONE). One of these replans uses the **know-one** operator (given in section 5.7) and involves asking fred whether or not he has SECTION-ONE; the other two use the **know-three** operator and involve telling fred the new descriptions for SECTION-ONE, (LEFT-OF PALM-BEACH-0), and (AROUND SWAMP-0), respectively. However, doris will never construct a cost estimate for the latter, since she uses her task planning communicative posture settings to choose just one possible replan for each operator at each level, and the former has a better evaluation. There are no possible replans that doris can use for her top level goal, (KNOW FRED ROUTE), because she knows of only one way to fulfill it.

Let us construct a cost estimate for each of the possible continuations in turn. The cost of the repair is the cost of the top level goal (6) plus the number of prerequisites that would be pushed (2) plus the number of levels from the top at which they would be pushed (3), or 11. The cost

of the replan using **know-one** is the cost of the top level goal (6) minus the cost of the current plan for the goal to be replanned (2) plus a penalty because the plan's relaxable constraints are not satisfied (500) plus a replanning threshold for a high risk agent replanning two levels from the top level goal (11 plus -1.6(2)), plus the cost of the replacement plan (the number of goals in the replacement, namely 0), or 511.8. The cost of the replan using **know-three** with the description (LEFT-OF PALM-BEACH-0) is the cost of the top level goal (6) minus the cost of the current plan for the goal to be replanned (2) plus the constraint penalty (0) plus a replanning threshold (11 plus -1.6(2)) plus the cost of the replacement plan (the number of goals in the replacement, namely 1), or 12.8.

There are several improvements that could be made to the way in which cost estimates are constructed. It would be better if the estimates for a new planning step were to take into account the prerequisites of the plan involved and check for prior occurrences of the same goal. For instance, if a new planning step involves the goal (KNOW FRED SWAMP-0), and that goal is already known to be difficult, then the cost estimates ought to reflect that. The fact that one communicative posture parameter, plan commitment, controls both factors of the replanning threshold makes the system unable to generate some plausible dialogues. We will have more to say about these dialogues in section 8.8.

7.4 The Layered Architecture

In section 2.3.3.1, we described how MOLGEN's layered architecture is structured so that the problem of scheduling tasks in a problem solver is broken down into smaller and smaller pieces using the problem solving techniques themselves. Stefik devised this technique because in ordinary agenda-based problem solving, the interpreter which selects tasks to be scheduled is hard to examine and change. The same criticism can be made of Moore's planner with respect to the recovery strategies, since it inflexibly runs procedures which decide whether or not a recovery strategy is appropriate at the time in which they are invoked. The criteria for deciding are simply hard-wired into the code; this is analogous to the situation Stefik describes where the problem solver's interpreter has domain knowledge prioritising the scheduling of tasks, but the structure of that knowledge is difficult to examine. The layered architecture which we describe next, however, unfolds the system's reasons for applying recovery heuristics where it does and makes it easier to vary their application based on the current context. It also makes it easier to add new recovery heuristics, since that entails adding new modular pieces to the architecture rather than changing the structure of a large chunk of the code. As a side effect, where before the two recovery heuristics that were implemented, replanning and repair, were very different

kinds of procedures invoked under different conditions, the layered planner elevates them to be comparable objects. The layered architecture can generate a whole range of dialogues which a planner built like Moore's could not, simply because its flexibility allows different agents to make different decisions about what to do next in a particular situation. This both gives a clearer picture of why things happen when they do and makes experimentation with different combinations of communicative posture settings and recovery strategies easier.

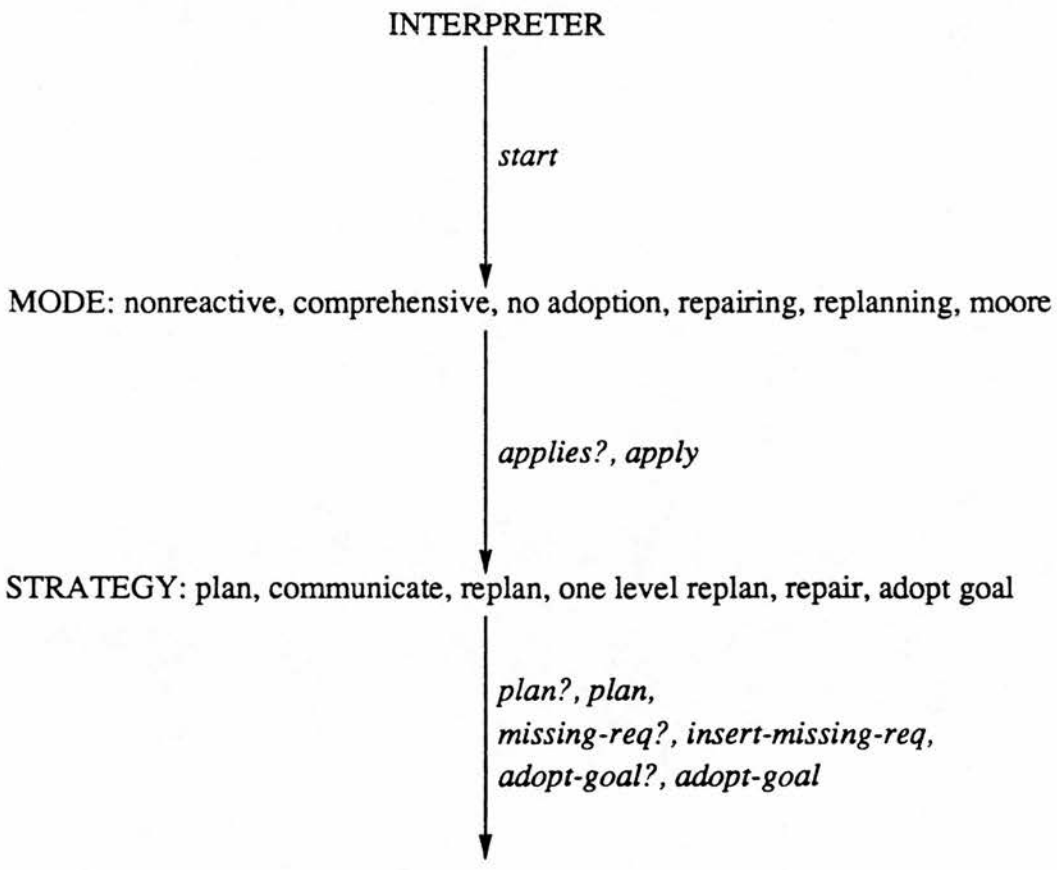
7.4.1 The Organisation of the Layers

Our system consists of four layers of planning operators:

> **The Domain Space** contains what would normally be the operators in a planning system for the domain. The mechanics of these operators have been described in detail in section 5.7; what makes them different from normal planning operators is that instead of being manipulated as data structures by planning routines, they work by exchanging messages with the plan strategy (among others). The operators at this level utilise the agent's knowledge about the domain, including knowledge about both the task and how dialogues are conducted.
>
> **The Strategy Space** contains operators corresponding to each of the possible ways of continuing from the current point in the plan. This involves normal planning and execution as well as all of the recovery strategies. This space is not related to MOLGEN's strategy space, but takes its name from the term "recovery strategy".
>
> **The Mode Space** controls which set of strategies are accessible to the agent and how a strategy is chosen. The simplest modes always give the agent access to the same set of strategies, but more complex modes can switch among sets depending on the current situation. For many applications one possible mode would suffice; in this case, we have several corresponding to different ways that people make strategy decisions in the human dialogues. This space is roughly analogous to MOLGEN's strategy space since it meta-plans the techniques used in continuing from the current point in the plan.
>
> **The Interpreter** simply chooses one of the mode operators to execute. In the JAM system, different modes correspond to different agents' ways of doing things, so the interpreter simply checks to see which agent it is for and invokes the appropriate mode. Many systems might only need one mode (combining the mode level and the interpreter) to bring about reasonable behaviour. Where other domains inspire more complex control at the higher levels, the interpreter can be as complex as required and even higher levels of the system may be added.

A diagram of the system is given in figure 7.2. This section describes in more detail the operators we have chosen for each layer and the messages that are passed between the layers. When we say that an operator "does" some action upon receiving a message, we mean that the sending of that message causes a particular procedure to be executed which does the action. Agents know which procedures to invoke because each agent has a data structure which associates operators and message names with the name of the procedure to run when that operator receives that message.

Figure 7.2: The System Architecture



7.4.2 The Domain Space

The domain space of the planner consists of operators which involve knowledge about dialogue and the domain. There are seven in all, and details about them are given in section 5.7. Here we repeat the English glosses of what the plan operators do:

Route: To get some agent to know the route on the map, get some other agent to describe all the sections of it in order.
Know-One: To get to know whether or not some agent knows some concept, ask that agent yes or no.
Know-Two: To get the self to know a concept, get the self to know some description of the concept.
Know-Three: To get the partner to know a concept, get the partner to know some description of the concept.
Know-Description-One: To get an agent to know some description of a concept, get some agent to tell the agent a description of the concept.
Know-Description-Two: To get to know some description of a concept, get the agent to ask another agent to describe the concept.
Nested-Know-One: To get the partner to know that the self has a concept, close the conversation about the concept.
Nested-Know-Two: To get to know that the partner has a concept, close the conversation about the concept.

Chapter 5 only described these operators as they could be used in any planner for this domain, but in the JAM system, these operators must work in the message passing architecture. Since the domain space is the bottom layer of the architecture, these operators do not send any messages. The messages which they can receive occur in pairs. The first message of each pair, postfixed with a question mark (?), asks the domain operator if it can be used for the associated strategy, and the second message asks the domain operator to apply itself according to that strategy:

Plan?: Asks the domain operator whether it can be used to plan the given goal. If it can, the operator stores in its register the most highly evaluated instantiation of itself which applies and returns the evaluation; otherwise, it returns nil.
Plan: Tells the domain operator to retrieve the plan which is stored in its register and push it on to the plan tree as a solution to the current goal. Additionally, the name of the domain operator is attached to the current goal for future reference.
Missing-req?: Asks the domain operator, which was used to plan the given node in the plan tree, whether any of its prerequisites are not yet fulfilled. This message is only used when the entire plan has been executed but the goal has not succeeded. If some prerequisites have not been fulfilled (i.e., as a result of an inappropriate high risk communicative posture), then the domain operator stores them and returns t; otherwise it returns nil.
Insert-missing-req: Tells the domain operator to retrieve the missing prerequisites which are stored in its register and to push them onto the plan tree at the current point.
Adopt-goal?: Asks the domain operator to try to recognise (in a limited way) the partner's current goal. This message is only sent when the agent has just reached a frame initiated by the partner or when the agent is working on a goal which it is acquired by using goal adoption. The recognition is done by having the domain operator see if it contains an

equivalent frame if it instantiates itself "from the partner's perspective". If the frame does occur in the operator, then the operator stores the instantiation of its effect (e.g., the partner's current bottom level goal) and returns t. Otherwise, the operator returns nil.

Adopt-goal: Tells the domain operator to retrieve the partner's current bottom level goal and push it as a sister node to the node in the plan tree which holds the frame which the partner initiated to further that goal.

7.4.3 The Strategy Space

The strategy space contains operators which embody each of the recovery heuristics plus normal planning and execution of dialogue games:

Plan: Considers ways to continue from the current situation by planning the current goal. *Applies?* returns a numerical value or nil.

Communicate: Considers ways to continue from the current situation by communicating with the partner, either by initiating a game which some other strategy has suggested, or by continuing a game initiated by either the self or the partner. *Applies?* returns t or nil. If the communication requires a response from the partner, *apply* returns "swap", telling the agent to relinquish control of the dialogue after this step.

Replan: Considers ways to continue from the current situation by replanning any one of the goals in the plan which have not yet been fulfilled. *Applies?* returns a numerical value or nil.

One Level Replan: Considers ways to continue from the current situation by replanning from a given goal in the plan which has not yet been fulfilled. This strategy operator is only used by modes which place orderings on the application of recovery strategies, like the mode reconstructing Moore's work. *Applies?* returns a numerical value or nil.

Repair: Considers ways to continue from the current situation by finding prerequisites to the current goal's plan which have not been fulfilled due to a communicative posture that was too high and fulfilling them. *Applies?* returns a numerical value or nil.

Adopt Goal: Considers ways to continue from the current situation by adopting what the agent thinks the partner's goal is in initiating a game which is currently in progress. *Applies?* returns t or nil.

These operators are controlled by two messages which are sent from mode space operators: *applies?* to query whether or not they are useful in the current situation, to which they return an evaluation, t, or nil, and *apply* to tell them to execute themselves. They, in turn, may need to communicate with the domain space operators, asking for information about the best way to apply themselves or passing on the call for execution. **Communicate** does not need to access any domain level operators, so it does not send out any messages. **Repair** uses *missing-req?* and *insert-missing-req*, and **adopt-goal** uses *adopt-goal?* and *adopt-goal.* Since **plan**, **replan**, and **one level replan** all involve the same basic operation but on different outstanding goals, they all use the *plan?* and *plan* messages but with different goals as the argument. **Plan** and **one level replan** work on the current bottom level goal of the plan. **Replan** works on each of the open goals in the plan successively.

When the strategy operators receive *applies?* messages, they return values which are used by the system to determine which one of the strategy operators to apply. **Communicate** and **adopt-goal** return only t or nil, indicating whether or not these strategies apply. The other strategies, **plan**, **replan**, **one level replan**, and **repair**, return a numerical value reflecting the estimated relative costs of completing the entire task if that strategy is applied next.

7.4.4 The Mode Space

The mode space controls the set of recovery strategies that is available to the agent at the present time and how the agent chooses to continue from the current situation. The modes range from the **nonreactive** mode, which is a simple plan/execute loop, to the **comprehensive** mode, which can employ any of the recovery strategies. The mode operator chooses one of the strategy operators it knows about and sends it a message to execute itself. The mode operators themselves can receive only one message from the interpreter, *start*. The mode operators are as follows:

Nonreactive: Simply do normal planning, communicating as soon as the need arises or a response to the partner is required. This mode allows us to check the effects of each recovery strategy without interference from this agent.

Comprehensive: Consider every possible continuation from the current situation and choose the least cost one, except always communicate as early as possible and always goal adopt. This mode is as close as we can get to a completely rational mode without improving the evaluation function.

No-Adoption: Work in exactly the same way as the **comprehensive** strategy, except do not ever adopt the partner's goals.

Repairing: Communicate as early as possible, always goal adopt, and only consider repair as a recovery strategy. This mode is useful for investigating repair.

Replanning: Communicate as early as possible, always goal adopt, and only consider replanning as a recovery strategy. This mode is useful for investigating replanning.

Moore: Reconstruct the behaviour of Moore's system as far as possible. In particular, go through the recovery strategies in the strict order of any possible repairs and then any possible replans. In addition, always adopt the partner's goals whenever possible and respond to the partner's communications before anything else (be cooperative).

Mode operators choose a strategy to employ by either having a preference for a particular strategy, as in, for instance, always completing communications first, or by considering a number of strategies and choosing the one which requires the least amount of effort. A totally rational mode would involve checking all the strategies and always applying the lowest cost one. However, writing an efficient evaluation function that covers all of the different strategies can be difficult. We have left our evaluation incomplete (so that some strategies, when polled only return t or nil to show whether or not it is possible to apply them) because we are primarily concerned with modelling human planning abilities, and it is not at all clear that the

typical resource-bounded human agent can evaluate his or her options so rationally. Our mode operators are intended to investigate the different possibilities open to agents and to reflect the surface-level behaviour from the human corpus of task-oriented dialogues. In addition, one operator, **Moore**, reconstructs the ordering for recovery strategies given in Moore's thesis [Moo90],[5] thus modelling the behaviour of her planner.

7.4.5 The Interpreter

This space consists simply of one operator which can receive one message, *start*, from the chair program which swaps between the two agents. At the moment, so that we can test the various modes by matching them up in different combinations, we have a number of different interpreters. Each agent uses a different one. The simplest interpreters run continually in one mode, employing a static set of recovery strategies, while more complex interpreters swap between modes based on whether or not recovery is necessary and whether or not the partner is attempting the recovery.

7.5 Worked Examples

This section describes how the JAM system uses the layered architecture to plan, execute, and recover from failures by giving two worked examples. It is not possible to specify one algorithm which all of the agents use in deciding what actions to take, because the algorithm depends on which interpreter (and hence which mode level operator) an agent uses. The first worked example shows the initial extract of a dialogue between two particular agents, carol and tom, and demonstrates communication, planning, repair, and goal adoption. This example both expands step six of the algorithm which is given in section 6.4.3 (refined from section 5.11) and fills in the gaps left in the worked example of section 6.4.4. The second example shows a simple case of replanning early in a dialogue.

7.5.1 Communication, Planning, Repair, and Goal Adoption

In the following example, carol and tom are both agents which use an interpreter which only calls the **comprehensive** mode and which both have high risk difference and context articulation posture settings and low risk settings for the remaining parameters. When carol is awakened by the chair program, she reads her mail, discovers the message which tells her to

[5] Reinstantiation is omitted for reasons given in section 4.2.5.

initialise her plan, and creates a plan tree with the top level goal of (KNOW TOM ROUTE), as indicated in section 5.11. Then she sends a *start* message to her interpreter (**comprehensive-interpreter**), which begins a loop that continually sends *start* messages to the **comprehensive** mode level operator until that operator returns the value 'swap, indicating that carol requires tom to respond to one of her utterances, or until that operator returns nil, indicating that it has reached the top of the plan with no more unfinished children and so carol has nothing to do. In either of these cases, carol's interpreter returns control to the chair program, giving up her turn.

In this case, the **comprehensive** mode has not just returned 'swap and carol does have a goal on which to work, and so her interpreter sends a *start* message to the **comprehensive** mode. The **comprehensive** mode first updates the currency pointer by the mechanism explained in section 5.5. Then it sets out to determine which strategy level operator to use, beginning by sending the *applies?* message to the **communicate** strategy.

The **communicate** strategy looks at the current node in the plan to determine whether or not there is anything to say stored in that node (i.e., that node contains a frame which has an utterance by carol which has not been said). In this case, the current node in the plan is a goal, so the **communicate** strategy returns nil to the **comprehensive** mode.

If the **communicate** strategy had returned t, then the **comprehensive** mode would have sent it an *apply* message, which would have sent the new utterance to the partner. Then the **comprehensive** mode would have sent an *applies?* message to the **goal-adopt** strategy, since this strategy sometimes applies when the partner has just initiated a dialogue game and this is the case only if the agent has had to respond, since there are no one move dialogue games in the current system. Then in turn if the **goal-adopt** strategy had returned t, the **comprehensive** mode would have sent it an *apply* message, making it adopt whatever goal was inferable from the partner's action. However, in this case the **communicate** strategy returned nil, and so the **comprehensive** mode moves on to look for other strategies to apply.

Next, the **comprehensive** mode sends the *applies?* message to the **plan** strategy, which indicates that the **plan** operator should determine whether or not it can do anything in the current situation. The **plan** strategy responds by sending the *plan?* message to each one of the domain level operators in turn. First the **plan** strategy sends the *plan?* message to the **route** operator. This makes the **route** operator check to see if it applies in the current situation. The mechanism by which domain level operators do this is given in section 5.8. In general, the operator gathers together all possible instantiations which match the current situation and evaluates each of them according to the task planning communicative posture parameters as described in section 5.12.1; then the operator stores the list of goals and frames

(including operations and, depending on communicative posture, prerequisites) constituting the most highly evaluated plan in its register, and returns its estimated cost, as described in section 5.9. In this case, there is just one possible instantiation of the operator which matches the current situation (a plan which involves getting tom to know all four sections of the route), and so the **route** operator stores a list of four goals in its register which correspond to getting tom to know how to get around the four sections of the route. Then the **route** operator returns the estimated cost of this plan to the **plan** strategy.

Next the **plan** strategy sends the *plan?* message to each of the other domain level operators. However, none of these operators match the current situation, and so they all return nil to the **plan** strategy.

Next, the **plan** strategy must decide which of the possibilities offered by the domain level operators is most desirable. It does this by selecting the one with the least cost, **route**, storing the name of the operator in its register, and returning that cost to the **comprehensive** mode.

So far, the **comprehensive** mode has worked towards determining the best strategy to apply by querying two of its five strategies about what they can do in the current situation. There are three remaining: **replan**, **repair**, and **adopt-goal**. It sends the *applies?* message to each of these strategies in turn. **Replan** returns nil because there are no open goals in the plan which could be replanned; so far there is only one goal on the tree, and it has not yet been planned at all. **Repair** returns nil because there have not been any failures to repair as yet.

Now that the **comprehensive** mode has evaluated all possible actions, it chooses whichever one has the least cost. In this case the only possible course of action is to use the **plan** strategy. Therefore the **comprehensive** mode sends the **plan** strategy the *apply* message. The **plan** strategy looks at its register to discover that the **route** operator was the best one that applied in the current situation, and so it sends the **route** operator a *plan* message. This makes the **route** operator push the plan stored in its register onto carol's plan tree underneath the current goal. This gives her the following plan tree:

```
open: (know tom route)
   unplanned: (know tom section-one)
   unplanned: (know tom section-two)
   unplanned: (know tom section-three)
   unplanned: (know tom section-four)
```

Carol's current point in the plan is the top level goal. Then the **route** operator returns t to the **plan** strategy, which returns t to the **comprehensive** mode, which returns t to the interpreter. At this point, the interpreter has completed one cycle. Since it did not receive a 'swap signal (indicating that carol needs a response from tom) or nil (indicating that the **comprehensive**

mode didn't find anything to do), it sends a *start* message to the **comprehensive** mode. The mode begins by updating the currency pointer in carol's plan so that the current goal is (KNOW TOM SECTION-ONE), since that is the leftmost unfinished child of the goal on which she had been working. Then, it begins to determine what strategy level operator to apply by sending the *applies?* message to the **communicate** strategy. The **communicate** strategy returns nil because the current point in the plan is a goal and hence there is nothing for carol to say. Next the **comprehensive** mode sends the *applies?* message to each of the **plan**, **replan**, and **repair** strategies in turn. Once again, the **replan** and **repair** strategies return nil because they do not apply, and the **plan** strategy returns a cost which is associated with the plan of getting tom to know the first section of the route by getting him to know some description of it by using the **know-three** domain level operator. As in the earlier case of planning, the **comprehensive** mode sends an *apply* message to the **plan** strategy, which sends an *apply* message to the **know-three** operator, which makes that operator push its plan onto carol's plan tree, giving it the following structure:

```
open: (know tom route)
   open: (know tom section-one)
      unplanned: (know tom (describes section-one
                             (between palm-beach-0
                                      swamp-0)))
   unplanned: (know tom section-two)
   unplanned: (know tom section-three)
   unplanned: (know tom section-four)
```

Then, as in the earlier case, the value t is returned up all the levels of the system to the interpreter, and the cycle begins again. Next by exactly the same process, but using the **know-description-one** operator, the plan tree is extended as follows:

```
open: (know tom route)
   open: (know tom section-one)
      open: (know tom (describes section-one
                        (between palm-beach-0
                                 swamp-0)))
         unexecuted: (12 inform carol
                         tom
                         (describes section-one
                           (between palm-beach-0
                                    swamp-0))
                         ?var10 ?var11)
   unplanned: (know tom section-two)
   unplanned: (know tom section-three)
   unplanned: (know tom section-four)
```

At this point carol is ready to make an utterance. This time when the **comprehensive** mode sends an *applies?* message to the **communicate** strategy, the strategy finds that there is a

new utterance to say, which is the initiating move of the current frame. It therefore returns t to the **comprehensive** mode, and the mode sends it an **apply** message, making it send the utterance to tom:

carol makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*carol says to tom: the first section of the route goes between the palm beach and the swamp.*

Then the **communicate** strategy returns the value 'swap to the **comprehensive** mode, indicating that the utterance requires a response before further action can be taken (as is the case with every dialogue move which does not complete a game). The **comprehensive** mode, because it receives the value 'swap, passes that value up to the interpreter, which cedes control to the chair program. This concludes one turn in the dialogue, and next the chair program gives control to tom.

Tom, upon being awakened, discovers the message which carol has left in his mailbox, reads it, updates his beliefs in line with it, and adds the dialogue game which it initiates to his discourse plan. Since he has no goals, this game is the only part of his plan. Then he sends a *start* message to his interpreter, **comprehensive-interpreter**. Since his interpreter has not just received the value 'swap from a mode level operator and because there is an unfinished node in the plan (namely, the initiated dialogue game), it sends a *start* message to the **comprehensive** mode, which first moves the currency pointer to the current goal in the plan to the initiated dialogue game. Then the **comprehensive** mode begins determining which strategy to apply by sending the *applies?* message to the **communicate** strategy. The **communicate** strategy, seeing that the current node is a game which is waiting for a response, returns t to the **comprehensive** mode. Then the **comprehensive** mode immediately sends an *apply* message to the **communicate** strategy. This causes the **communicate** strategy to construct an appropriate responding move,[6] send the utterance to carol, update tom's beliefs in line with the response, and return control to the **comprehensive** mode:

tom makes an inform(r) move with content not-ok	*tom says to carol: i do not understand.*

Unlike with the previous utterance, the returned value is t, not 'swap, because this move closes the dialogue game and tom is free to continue his turn. Because tom has just received a message from the partner initiating a new game and has responded to it, the **comprehensive** mode next sends an *applies?* message to the **goal-adopt** strategy, which causes it to look at the current situation for evidence of what the partner's goals are.

[6] It might perhaps be more appropriate to have the **communicate** strategy construct the new utterance when it is sent the *applies?* message and store it in a register for later use, since the value returned by the *applies?* message indicates that the agent has something to say. This requires only a very small modification to the current arrangement.

At this point, tom's currency pointer rests on the game which he has just completed. The **goal-adopt** strategy sends an *adopt-goal?* message to each of the domain level operators in turn, which asks them whether or not the current node is consistent with use of the operator. A domain operator matches for the purposes of goal adoption if the current game or goal unifies with one of the operations on the domain operator's list in such a way that if tom substitutes himself for carol, carol for himself, what he thinks carol believes for what he believes, and what he thinks carol thinks he believes for what he thinks carol believes in a use of the operator, the nonrelaxable constraints of the operator are satisfied. In addition, the new goal which is the effect of such a match must not already be an ancestor of the current node in tom's discourse plan, since it would not be sensible to require the goal to be solved as just one part of solving itself. All of the domain level operators return nil except for **know-description-one**; this operator stores in its register the goal node which represents the effect of carol's plan and returns t. Then the **goal-adopt** strategy sends it an *adopt-goal* message. The **know-description-one** operator retrieves the goal node which it has stored, pushes that effect onto the plan tree as the parent of the current node, and changes the former parent of the current node to be the parent of the new node. Since tom does not believe that the goal is already fulfilled, the new node is created with the status 'open, and since it arose as the result of goal adoption, it is marked with a special flag which allows tom to adopt another goal which might explain carol's having that goal. In addition, the new node is marked with a flag that makes it so that tom does not consider replanning the goal; any change in plan must come from carol. Then, the goal having been adopted, the **know-description-one** operator returns t to the **goal-adopt strategy**.

Since a goal was successfully adopted, as explained in section 7.2.5 the **goal-adopt** strategy recursively sends itself an *applies?* message but using the new node instead of the current node to determine what purpose the partner might have in wanting the current goal to be fulfilled. In this invocation it determines that carol used the **know-three** operator and adopts the goal (KNOW TOM SECTION-ONE). Then in the next invocation it determines that carol used the **route** operator and adopts the goal (KNOW TOM ROUTE). At this point, the recursion stops, since the **goal-adopt** strategy can not determine any higher level reason for carol's behaviour. At the end of the recursion, tom's discourse plan has the following form

```
open: (know tom route)
     open: (know tom section-one)
        open: (know tom (describes section-one
                                (between palm-beach-0 t114)))
           executed: (12 inform carol tom
                                 (describes section-one
                                  (between palm-beach-0 t114))
```

```
not-ok ?var11)
```

and his currency pointer rests on the game at the bottom. The **goal-adopt** strategy finally returns t to the **comprehensive** mode.

The **comprehensive** mode next updates the currency pointer, placing it on the goal (KNOW TOM (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 T114))). Then it polls the strategies with the *applies?* message to determine which one is best in the current situation. **Communicate** and **goal-adopt** return nil because the currency pointer is not resting on a dialogue game or a goal which has been adopted from the partner, and **plan** returns nil because there is a failure from which the agent must recover before proceeding. **Replan** returns nil because tom has adopted all goals in the current line of work from carol, and tom does not consider changing any plans which carol has initiated. **Repair**, however, is applicable in this situation.

When the **comprehensive** mode sends the *applies?* message to the **repair** strategy, **repair** looks for any prerequisites to the plan which might have been omitted by retrieving the name of the operator which was used in constructing the plan and working out what the prerequisites are but without eliminating any which might have been ignored due to the agents' communicative posture settings. In this case, **repair** finds two missing prerequisites, (KNOW TOM PALM-BEACH-0) and (KNOW TOM T114). Only the one regarding t114 is unfulfilled, and so it stores it in its register and returns the cost of the repair to the **comprehensive** mode. The **comprehensive** mode then sends it an *apply* message, since it is the only strategy which is applicable. The **repair** strategy then adds the prerequisite to tom's plan tree under the current node. In addition, it adds a goal that, once the repair is done, carol know that it is done (i.e., carol know that tom knows the information that she has been trying to convey). After the repair, tom's discourse plan has the following structure:

```
open: (know tom route)
   open: (know tom section-one)
      open: (know tom (describes section-one
                          (between palm-beach-0 t114)))
         executed: (12 inform carol tom
                               (describes section-one
                                (between palm-beach-0 t114))
                               not-ok ?var11)
         unplanned: (know tom t114)
      unplanned: (know carol (know tom
                                    (describes section-one
                                     (between palm-beach-0 t114))))
```

At this point in the example, we have demonstrated the use of every recovery strategy except replanning. The remainder of this dialogue can be found in section 8.3.

7.5.2 Replanning

The last section gave a worked example which used all of the different ways of continuing from the current situation in a dialogue except replanning. For completeness, this section gives an example in which the route giver, anne, decides to replan early in the dialogue. Anne and andrew are both agents which use an interpreter which only calls the **no-adoption** mode and which both have high risk difference, context articulation, and plan commitment posture settings and low risk settings for the remaining parameters. When anne is awakened by the chair program, she follows the same steps that carol followed in the last example in order to adopt the top level goal (KNOW ANDREW ROUTE). Then she sends a *start* message to her interpreter (**no-adoption-interpreter**), which begins a loop that continually sends *start* messages to the **no-adoption** mode level operator until that operator returns the value 'swap, indicating that control should be ceded to andrew so that he can make a required utterance. Anne and andrew have exactly the same behaviour as carol and tom initially, since even though they choose what to do differently, there is only one possibility at each of the initial steps. Thus, in exactly the same way as in the last dialogue, except using the **no-adoption** mode, anne and andrew have the following conversation:

anne makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*anne says to andrew: the first section of the route goes between the palm beach and the swamp.*
andrew makes an inform(r) move with content not-ok	*andrew says to anne: i do not understand.*

At this point, andrew's discourse plan has nothing but the completed game in it. In tom's case, he next used three instances of goal adoption to reconstruct carol's reasons for making the statement that she did. However, andrew uses the **no-adoption** mode, which does not allow him to adopt anne's goals. With no goals to satisfy, andrew has no work to do. His interpreter, since it is at the top of the discourse plan with no open goals, cedes control to anne.

When anne is awakened, she finds and interprets the message in her mailbox in the same way that carol does. At this point her discourse plan has the following structure:

```
open: (know andrew route)
   open: (know andrew section-one)
      open: (know andrew (describes section-one
                                  (between palm-beach-0 swamp-0)))
```

```
executed: (12 inform anne andrew (describes section-one
                                   (between palm-beach-0
                                    swamp-0)) not-ok ?var11)
unplanned: (know andrew section-two)
unplanned: (know andrew section-three)
unplanned: (know andrew section-four)
```

Since the dialogue game is finished, her interpreter updates the currency pointer to the goal above it and then sends an *applies?* message to each of the different continuation strategies in order to determine what to do next. **Communicate**, **goal-adopt**, and **plan** all return nil, since none of these strategies can be applied in the current situation. **Repair** constructs the repair involving the two goals (KNOW ANDREW PALM-BEACH-0) and (KNOW ANDREW SWAMP-0), records this choice, and returns the cost of the repair (11), all as detailed in the last example. **Replanning** is also applicable in this situation. when the **replanning** mode receives the *applies?* message, since the current node is an open goal it gathers together a list of the open ancestors of the current node (including that node itself) from top to bottom which are not marked with a special flag indicating that they may not be replanned. Then it works through this list in order, looking for the best replan to propose to the **comprehensive** mode. Anne's first open node, (KNOW ANDREW ROUTE), is marked as not replannable for efficiency reasons, since she only knows one plan for that goal. Therefore the **replan** strategy only investigates possibilities for the other two open nodes. Starting with the goal (KNOW ANDREW SECTION-ONE), it sends the *plan?* message to each of the domain level operators in turn, asking them to find their best replan for the node given anne's task planning communicative posture settings. Replans are found in the same way as original plans, except that an additional check is done to ensure that the new plan has not been tried before in the same situation[7] and the cost of the lowest cost replan so far is passed to the domain level operators so that they will not propose any replans which are not improvements.[8] In this case, two of the domain level operators return non-nil values: **know-one** proposes the plan of asking andrew whether or not he knows the concept section-one at a cost of 508.2, and **know-three** proposes the plan of substituting the description (describes section-one (left-of palm-beach-0)) at a cost of 9.2. Then the **replan** strategy broadcasts the *applies?* message to the domain level operators again using the goal (KNOW ANDREW (DESCRIBES SECTION-ONE (BETWEEN PALM-BEACH-0 SWAMP-0))). All of the operators return nil since they are not applicable.[9] At this point the **replan** strategy

[7] A replan is considered not to have been used before if not all of the operations of the replan are already daughters of the goal which is being replanned.

[8] This is necessary to avoid register collisions. If the same domain level operator were applicable to two different goals in the list, and the better replan were found first, it might be overwritten in the operator's register by the worse replan and lost. However, if the registers were written in a more sensible way, for instance, by allowing multiple entries in the register keyed on how each was determined, passing the lowest cost so far to the domain level operators would not be necessary.

[9] The know-one operator does not apply because it contains a nonrelaxable constraint which says that the

has exhausted the possible options. Since the lowest cost replan was the one which applied the **know-three** operator to the goal (KNOW ANDREW SECTION-ONE), it stores the name of the operator and the goal at which the replan would be done in its register and returns the cost, 9.2, to the **comprehensive** mode.

The **comprehensive** mode must now choose between repair at a cost of 11 and replanning at a cost of 9.2. It chooses the lower cost option, and thus sends an *apply* message to the **replan** strategy. This causes the strategy to mark as abandoned all of the nodes involved in the old plan for the goal node stored in its register (that is, all unfinished nodes below that goal node), move the currency pointer to that goal node, and send a *plan* message to the domain level operator named in its register, **know-three**. The **know-three** operator then pushes the new plan onto the discourse plan at that goal node in exactly the same way as it would do for a new plan.

At this point anne's decision to replan is complete. The remainder of the dialogue is given in section 8.6.

7.6 Conclusions

The JAM system improves upon previous methods for incorporating recovery strategies into planning systems. The major part of the improvement comes from making an estimate of the amount of effort which the task will require for each of the possible ways of continuing from the current situation and then always choosing the option which requires the least effort. As a side effect, replanning can be considered even in situations where no plan failure has occurred, as sometimes happens in human dialogues. A JAM dialogue in which this occurs is given in section 8.5. The JAM system also increases flexibility by making decisions about what recovery strategy to employ via a layered architecture. In this system, recovery strategies are meta-planning operators which manipulate the domain level operators and plan. In addition, higher level operators decide which recovery strategy to employ based on both their estimated costs and different partial orderings of the recovery strategies where no methods for calculating such costs are available. This arrangement is simple to modify, making it easier to investigate interactions among the recovery strategies and to simulate a range of human dialogues from the corpus.

Although the JAM architecture was developed primarily for use in simulation, it may also be useful for other purposes. Any planner which can operate in an uncertain environment must be

---

concept which andrew is to learn must be "describable", not a description itself.

able to invoke recovery strategies sensibly, and so might profitably be built using the layered architecture described. Moreover, work on recovery strategies and architectures which employ them could lead to a different approach to planning altogether. In uncertain environments where executing plans is not "expensive" (e.g., the objects involved are not easily damaged or are replaceable, and not much effort is needed to restore the world to a state in which a new plan for the same goal can be attempted), it may be rational for an agent to do minimal planning in the first instance and rely heavily on recovery strategies to guide it to a solution. In the human task-oriented dialogues, agents often initially leave out steps of their plans which turn out to be vital. Presumably they assume that their failures will be recoverable and they think that they will use less effort overall if they adopt this tactic. It may be that this behaviour will also prove more rational than relying on careful planning for some more traditional planning domains.

Chapter 8

Output from the JAM System

This chapter comments on a number of different dialogues which can be produced by the JAM system, all of which come about because of the same piece of information which the agents do not share. All of the dialogues cover the first section of map A in appendix A. This chapter consists mostly of annotated output from the simulation. Section 8.1 gives a dialogue extract in which both agents have low risk communicative posture settings for each of the implemented parameters (ontology, partner modelling, difference, context articulation, and plan commitment). To show the difference that the communicative posture settings make, section 8.2 gives a dialogue extract in which both agents have totally high risk communicative posture settings. The remaining sections give dialogue extracts which show different ways in which agents can recover from a plan failure which arises due to a high risk difference setting. Section 8.3 and section 8.4 give dialogue extracts in which repair is used by the route follower and the route giver, respectively. Section 8.5 gives a dialogue extract in which the route giver uses replanning at the lowest level goal to recover from the same failure. Section 8.6 contrasts two dialogues which differ only in the setting of the plan commitment parameter, which controls how much lower cost a replan must be in comparison to continuing from the current plan before the agent will choose to replan. Section 8.7 gives a dialogue extract in which the route follower begins with the top level goal and thus controls the course of the dialogue except where the route giver initiates a recovery. Finally, section 8.8 gives two dialogues which the JAM system can not generate and explains why. The extracts given in this chapter are representative of the dialogues which the system can generate but do not cover all of the forms of dialogues that can be generated. Extracts initiated using the "talk" procedure give the route giver the top level goal of having the partner know how to get around the route. This is the more usual arrangement in the corpus. Extracts initiated using the "rev-talk" procedure, such as that in

Figure 8.1: The JAM Dialogue Agents

	Role	Interpreter	Posture Settings				
			Ont.	Mod.	Diff.	Con.	Com.
mary	G	comprehensive	0	0	0	0	0
john	F	comprehensive	0	0	0	0	0
mac	G	nonreactive	1	1	1	1	1
jack	F	nonreactive	1	1	1	1	1
carol	G	comprehensive	0	0	1	1	0
tom	F	comprehensive	0	0	1	1	0
tim	G	nonreactive	0	0	1	1	0
doris	G	no-adoption	0	0	1	1	0
fred	F	no-adoption	0	0	1	1	0
andrew	F	nonreactive	0	0	1	1	0
anne	G	comprehensive	0	0	1	1	1
janet	G	comprehensive	0	0	0	1	1

section 8.7, give the route follower the top level goal of learning how to get around the route. Given which agent has the top level goal, the system generates different dialogues depending on the agents' communicative posture settings and choice of interpreter. The settings for each of the agents mentioned in this chapter are given in figure 8.1. In the figure, "role" is either G for a route giver or F for a route follower, indicating which map the agent has. The interpreters are those given in section 7.4.5. The communicative posture parameters are ontology, partner modelling, difference, context articulation, and plan commitment, respectively. In addition, one agent, andrew, has a slightly modified map which does not contain the waterfall to demonstrate the influence of the plan commitment parameter.

8.1 A Low Risk Dialogue

In the following dialogue, mary and john are both agents with low risk postures for each of the parameters that is implemented. This makes the agents explicitly open subdialogues about new topics, choose descriptions which are detailed and are as appropriate for the current partner as possible, and check the agent's knowledge about objects on the map before using descriptions which mention them. For this dialogue, mary begins with the goal that john know how to get around the route and john begins with no goals. First, since mary has a low risk context articulation setting, she opens the first topic to be discussed:

(talk *mary* *john*)
mary makes an open(i) move
with content "first section"
john makes an open(r) move
with content ok

*(talk *mary* *john*)*
mary says to john: i want to talk about the first section.
john says to mary: ok.

Next, mary chooses a particular description of this section of the route, which she will eventually realise as "between the palm beach and the swamp". She chooses this description because it is rated most highly according to the ontology and partner modelling parameters, as described in section 5.12.1. Since she has a low risk difference posture, she checks first to see whether or not john has these two map objects:

mary makes a yes-no(i) move with content "palm beach"	*mary says to john: do you have the palm beach?*
john makes a yes-no(r) move with content yes	*john says to mary: yes.*
mary makes a yes-no(i) move with content "swamp"	*mary says to john: do you have the swamp?*
john makes a yes-no(r) move with content no	*john says to mary: no.*

Here, mary has found a problem with her plan – in order to use the description she has chosen, she must get john to know about the swamp. She considers herself to have run into a plan failure at this point; her plan to get to know that john knows about the swamp by asking him a yes-no question has not worked. Rather than replan her description of the first section of the route, she decides to replan getting to know that john knows about the swamp, by telling him a description of it. First she focuses his attention on the swamp:

mary makes an open(i) move with content "swamp"	*mary says to john: i want to talk about the swamp.*
john makes an open(r) move with content ok	*john says to mary: ok.*

Then she chooses a description of the swamp which involves the waterfall and the palm beach, and tries to establish that john has these objects. However, she already knows that john has the palm beach, so she only needs to ask about the waterfall.

mary makes a yes-no(i) move with content "waterfall"	*mary says to john: do you have the waterfall?*
john makes a yes-no(r) move with content yes	*john says to mary: yes.*

At this point, mary knows that john has enough information to understand her description of the swamp, so she gives it:

mary makes an inform(i) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*mary says to john: the swamp is between the waterfall and the palm beach.*
john makes an inform(r) move with content ok	*john says to mary: ok.*

Furthermore, now mary knows that john has enough information to understand her description of the first section of the route.

mary makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*mary says to john: the first section of the route goes between the palm beach and the swamp.*
john makes an inform(r) move with content ok	*john says to mary: ok.*

At this point, mary and john are done speaking about the first section of the route, and mary begins again with the second section:

mary makes an open(i) move with content "second section"	*mary says to john: i want to talk about the second section.*
john makes an open(r) move with content ok	*john says to mary: ok.*
mary makes a yes-no(i) move with content "cliffs"	*mary says to john: do you have the cliffs?*
john makes a yes-no(r) move with content yes	*john says to mary: yes.*

The dialogue continues until they have completed the task. Note that mary does not initiate closing games for either the first section or the swamp, even though she explicitly opens dialogues about them. As we noted in section 5.4, opening and closing games are not symmetric. Opening games inform the partner about the current context and are used by agents with low risk context articulation settings whenever there is a major context shift, while closing games are used only when a recovery is complete which is not obvious to both agents and serve to coordinate the knowledge of the two agents. In this case, mary does not need to close the subdialogue about the swamp even though she had to replan because she knows that both john must understand the swamp and john must know that she knows he understands it. This comes about as a result of the belief propagation rules described in chapter 6; since mary has firmly established every possible prerequisite to the plan, and because she only knows about failures that have to do with prerequisites, she knows that john must understand the swamp. Furthermore, since john has observed the same dialogue as she has, she knows that john must know that she knows he understands it. Of course, in natural dialogue agents tend not to be so sure about their beliefs, so they might well explicitly close the topic "just in case". We leave it to further work to produce such a dialogue, since having a system allow both it and the john and mary dialogue given would require reasoning about degrees of belief.

8.2 A High Risk Dialogue Without Recovery

In the previous section, we gave a dialogue in which low risk communicative posture yielded a fairly trouble-free but inefficient dialogue. This section contains a dialogue in which the agents use high risk postures and encounter a plan failure. However, in this dialogue we have

handicapped the agents by not providing them with any means of recovering from failures whatsoever; mac and jack only know how to create the original plan and communicate, and neither agent tries to recognise the partner's goals. Note that mac chooses a simpler description of the first section of the route than mary did because mac uses high risk task planning parameters while mary's parameter settings are entirely low risk.

(talk *mac* *jack*)	*(talk *mac* *jack*)*
mac makes an inform(i) move with content (describes "first section" ("around" "swamp"))	*mac says to jack: the first section of the route goes around the swamp.*
jack makes an inform(r) move with content not-ok	*jack says to mac: i do not understand.*

At this point in the dialogue, mac has goals which are not fulfilled but has no strategies which she can apply to them, so she considers herself to be "stuck". On the other hand, jack has no goals at all, and so says nothing.

8.3 Repair by the Route Follower

In this dialogue, high risk agents successfully recover from a plan failure. Carol and tom both know all of the implemented recovery strategies and always adopt each other's goals. This leads to a dialogue in which tom, the route follower, diagnoses the same problem as occurs in the dialogue between mac and jack. He goes on to choose the recovery strategy which he believes is best suited to the situation, in this case repair, and then tells carol that recovery is complete. This is the same dialogue as was described in detail in section 7.5.

(talk *carol* *tom*)	*(talk *carol* *tom*)*
carol makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*carol says to tom: the first section of the route goes between the palm beach and the swamp.*
tom makes an inform(r) move with content not-ok	*tom says to carol: i do not understand.*
tom makes a wh-ask(i) move with content (describes "swamp" ?var356)	*tom says to carol: where is the swamp?*
carol makes a wh-ask(r) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*carol says to tom: the swamp is between the waterfall and the palm beach.*
tom makes a wh-ask(f) move with content ok	*tom says to carol: ok.*
tom makes a close(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*tom says to carol: i am done talking about the first section.*
carol makes a close(r) move with content ok	*carol says to tom: ok.*
carol says to tom: the first section of the route goes between the palm beach and the swamp.	*carol says to tom: the second section of the route goes between the waterfall and the cliffs.*
carol makes an inform(i) move with content (describes "second section" ("between" "waterfall" "cliffs"))	*tom says to carol: ok.*
tom makes an inform(r) move with content ok	

Because the system does not allow mixed initiative dialogues and because agents always answer questions put to them, exactly the same dialogue occurs no matter which high risk route giver is paired with tom. However, the route giver's record of the dialogue will differ depending on whether or not it adopts tom's goals; in this case carol, who does adopt tom's goals, has a deep tree representing why the dialogue has taken the course that it has up to the end of the extract given, as shown in figure 8.2.

Although tim, a high risk agent who knows no recovery strategies, has exactly the same dialogue with tom, his representation of it does not record why he thought tom initiated the dialogue games that he did, leaving him with a flat structure except for those things which he planned himself, as shown in figure 8.3.

As far as tim is concerned, the dialogue games initiated by tom are unrelated to his plan and he merely places them in the current place in the dialogue history, responds to them, and moves on.

Figure 8.2: Carol's Discourse Structure

```
open: (know tom route)
   succeeded: (know tom section-one)
      succeeded: (know tom (describes section-one
                              (between palm-beach-0 swamp-0)))
         executed: (815 inform carol tom
                        (describes section-one
                          (between palm-beach-0 swamp-0)) not-ok ?var352)
         succeeded: (know tom swamp-0)
            succeeded: (know tom (describes swamp-0 ?var361))
               executed: (821 wh-ask tom carol
                              (describes swamp-0 ?var361)
                              (describes swamp-0
                                (between waterfall-0 palm-beach-0)) ok)
         executed: (826 close tom carol
                        (describes section-one
                          (between palm-beach-0 swamp-0)) ok ?var365)
   open: (know tom section-two)
      open: (know tom (describes section-two
                        (between waterfall-0 cliffs-0)))
         initiated: (836 inform carol tom
                        (describes section-two
                           (between waterfall-0 cliffs-0))
                        ?var366 ?var367)
   unplanned: (know tom section-three)
   unplanned: (know tom section-four)
```

Figure 8.3: Tim's Discourse Structure

```
open: (know tom route)
   succeeded: (know tom section-one)
      succeeded: (know tom (describes section-one
                              (between palm-beach-0 swamp-0)))
         executed: (12 inform tim tom
                       (describes section-one
                          (between palm-beach-0 swamp-0)) not-ok ?var21)
         executed: (18 wh-ask tom tim
                       (describes swamp-0 ?var30)
                       (describes swamp-0
                          (between waterfall-0 palm-beach-0)) ok)
         executed: (19 close tom tim
                       (describes section-one
                          (between palm-beach-0 swamp-0)) ok ?var34)
   open: (know tom section-two)
      open: (know tom (describes section-two
                          (between waterfall-0 cliffs-0)))
         initiated: (28 inform tim tom
                        (describes section-two
                           (between waterfall-0 cliffs-0)) ?var35 ?var36)
   unplanned: (know tom section-three)
   unplanned: (know tom section-four)
```

8.4 Repair by the Route Giver

In the previous example, the route follower, tom, was able to initiate recovery from the failure because he could recognise carol's plan and therefore knew what prerequisites could have failed. Of course, tom was in a better position to diagnose the error than carol would have been because the prerequisites had to do with tom's knowledge. If tom had decided not to try to recognise carol's goal, carol would have been forced to go through all of the prerequisites which her high risk difference posture had allowed her to skip over, checking that each one holds. Doris and fred are set up to produce the dialogue that results under these circumstances. Here, both fred and doris know how to repair and replan, but neither of them adopts the partner's goals, forcing the agent who originates a plan to initiate any recovery necessary because the plan fails.

The dialogue begins in exactly the same way as the last example:

(talk *doris* *fred*)	*(talk *doris* *fred*)*
doris makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*doris says to fred: the first section of the route goes between the palm beach and the swamp.*
fred makes an inform(r) move with content not-ok	*fred says to doris: i do not understand.*

However, fred passes up his chance to diagnose the failure and returns control to doris, forcing her to check each of the prerequisites in turn:

doris makes a yes-no(i) move with content "palm beach"	*doris says to fred: do you have the palm beach?*
fred makes a yes-no(r) move with content yes	*fred says to doris: yes.*
doris makes a yes-no(i) move with content "swamp"	*doris says to fred: do you have the swamp?*
fred makes a yes-no(r) move with content no	*fred says to doris: no.*

At this point doris has found a failure and works to correct it; in the map task dialogues, it suffices to satisfy prerequisites after the fact, and so she simply sets out to get fred to know about the swamp.

doris makes an inform(i) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*doris says to fred: the swamp is between the waterfall and the palm beach.*
fred makes an inform(r) move with content ok	*fred says to doris: ok.*

Just coincidentally, it was the last prerequisite which turned out to be the one that failed. If there were more prerequisites to check, doris, who does not make any assumption about

there being a minimal number of failures, would continue to check the rest. Unlike in the last dialogue, the agents now move directly to the next section of the route:

doris makes an inform(i) move with content (describes "second section" ("between" "waterfall" "cliffs"))	*doris says to fred: the second section of the route goes between the waterfall and the cliffs.*
fred makes an inform(r) move with content ok	*fred says to doris: ok.*

Doris does not initiate a closing game to establish mutual understanding of the first section for the same reason that mary in section 8.1 did not initiate a closing game to establish mutual understanding of the swamp. Since doris has established every possible prerequisite to the plan, and because she only knows about failures that have to do with prerequisites she knows that fred must understand the first section and must know that she knows it. Thus fred and doris are free to move on to subsequent sections of the route, and the dialogue continues in the same manner.

8.5 Replanning

All of the high risk examples given so far which have successfully recovered from problems with information about the first section of the route have used repair as the recovery strategy because they have had low risk plan commitment settings, which, in the situation which we are investigating, leads the agent to attempt repair first. In this section, we give a dialogue in which the route follower does not know any recovery techniques and the route giver, janet, has a high risk plan commitment setting, causing her to choose replanning instead of repair even before the entire initial plan has been executed. In the example, janet also has a low risk difference setting, and so she begins the dialogue by deciding to tell her partner, jack, to go between the palm beach and the swamp and asking jack about the map objects involved:

(talk *janet* *jack*)	*(talk *janet* *jack*)*
janet makes a yes-no(i) move with content "palm beach"	*janet says to jack: do you have the palm beach?*
jack makes a yes-no(r) move with content yes janet makes a yes-no(i) move with content "swamp" jack makes a yes-no(r) move with content no	*jack says to janet: yes.* *janet says to jack: do you have the swamp?* *jack says to janet: no.*

At this point, janet has discovered that her initial plan is not as good as she thought it would be. Since she has a high risk plan commitment setting, she estimates the cost of repairing her plan as higher than the cost of trying another plan which doesn't mention the swamp, and so

she chooses the description "to the left of the palm beach". Of course, since she already knows that jack has the palm beach, all she needs to execute of her new plan is the informing game:

janet makes an inform(i) move with content (describes "first section" ("left-of" "palm beach"))	*janet says to jack: the first section of the route goes to the left of the palm beach.*
jack makes an inform(r) move with content ok	*jack says to janet: ok.*

This behaviour is very common in human dialogues where agents use low risk difference settings, since it involves making use of one's model of the partner in order to come up with a description that the partner will be able to understand.

8.6 The Influence of Plan Commitment

In the last section, we gave an example of a high risk dialogue in which an agent used replanning to recover from a failure in the dialogue even before the entire plan had been executed. This was made possible by a low risk difference setting which gave the agent more information about the likely success of the main informing game of the plan ahead of time and a low risk plan commitment setting which made the agent choose to replan rather than repair the existing plan. Of course, it is perfectly possible for an agent to decide to replan even when it doesn't know exactly what the problem is; in cases where repair involves the effort of diagnosing the problem whereas replanning might not, this is a likely scenario. This section contrasts two dialogues in which the agents are identical except that the route giver in the first has a high risk plan commitment setting whereas the route giver in the second has a low risk setting. This makes the agent make different choices about when to replan or repair based on the estimated cost to complete the task given each strategy. A low risk plan commitment setting makes agents less likely to replan by making them more pessimistic about the expected cost of replanning; although all agents believe that replanning does entail some extra cost in terms of the effort needed to abandon the old plan and adopt the new one, agents who have low risk plan commitment settings believe that the extra cost is higher than their high risk counterparts. In both dialogues, the route follower, andrew, does not know any recovery strategies and has a slightly modified map which omits the waterfall.

In the following dialogue, the route giver, anne, has a high risk plan commitment setting. The dialogue begins in the usual way, and andrew, since he does not know any recovery strategies, simply tells anne that he does not understand her statement and leaves her to diagnose the problem:

(talk *anne* *andrew*)	*(talk *anne* *andrew*)*
anne makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*anne says to andrew: the first section of the route goes between the palm beach and the swamp.*
andrew makes an inform(r) move with content not-ok	*andrew says to anne: i do not understand.*

Anne decides that the lowest cost way to finish the task is to try another description without diagnosing the problem with her old plan:

anne makes an inform(i) move with content (describes "first section" ("left-of" "palm beach"))	*anne says to andrew: the first section of the route goes to the left of the palm beach.*
andrew makes an inform(r) move with content ok	*andrew says to anne: ok.*

In this case, her high risk plan commitment setting "pays off" because her new plan works, and the agents go on to discuss the next section of the route. In particular, the agents never discover that the waterfall is not present on both maps.

In the next dialogue, the route giver, carol, decides to repair the initial plan because she has a low risk plan commitment setting. The dialogue begins normally:

(talk *carol* *andrew*)	*(talk *carol* *andrew*)*
carol makes an inform(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*carol says to andrew: the first section of the route goes between the palm beach and the swamp.*
andrew makes an inform(r) move with content not-ok	*andrew says to carol: i do not understand.*

Next, carol diagnoses the problem by beginning a repair:

carol makes a yes-no(i) move with content "palm beach"	*carol says to andrew: do you have the palm beach?*
andrew makes a yes-no(r) move with content yes	*andrew says to carol: yes.*
carol makes a yes-no(i) move with content "swamp"	*carol says to andrew: do you have the swamp?*
andrew makes a yes-no(r) move with content no	*andrew says to carol: no.*

In this case, carol decides that the cost of telling andrew the location of the swamp is less than the cost of replanning the entire first section of the route, and so she tells andrew where the swamp is:

carol makes an inform(i) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*carol says to andrew: the swamp is between the waterfall and the palm beach.*

Unfortunately for carol, we have modified andrew's map so that he doesn't have the waterfall, and so her plan is unsuccessful:

andrew makes an inform(r) move with content not-ok	*andrew says to carol: i do not understand.*

Carol once again decides not to replan at the top level, but this time she decides to replan the information about the swamp rather than diagnose what went wrong with her last plan:

carol makes an inform(i) move with content (describes "swamp" ("left-of" "palm beach"))	*carol says to andrew: the swamp is to the left of the palm beach.*
andrew makes an inform(r) move with content ok	*andrew says to carol: ok.*

Because carol has diagnosed the problem with her plan for getting around the first section of the route, she knows that andrew has the palm beach on his map and thus chooses the description "to the left of the palm beach" for her new plan for explaining where the swamp is. At this point, carol has supplied andrew with enough information to get around the first section of the route.

8.7 A Dialogue Initiated by the Route Follower

In all of the examples which we have given so far, the route giver has the top level goal of getting the route follower to know how to follow the route. Since our goal adoption strategy does not allow an agent to initiate plans for subgoals about which the partner has not already spoken, this makes the route follower a fairly passive agent that can only initiates dialogue games when it discovers a plan failure. However, it is also possible to run the system so that the route follower has the top level goal of getting to know the route and the route giver is the more passive agent. In the following dialogue, both agents have high risk communicative posture settings for all implemented parameters, know about both repair and replanning, and adopt goals wherever possible.

The dialogue begins with tom asking carol for the information he wants:

(rev-talk *carol* *tom*)	*(rev-talk *carol* *tom*)*
tom makes a wh-ask(i) move with content (describes "first section" ?var198)	*tom says to carol: where is the first section?*

Except for the initial questions about each section of the route which get each dialogue topic established, the dialogue proceeds in much the same way as the other dialogue between carol and tom:

carol makes a wh-ask(r) move with content (describes "first section" ("between" "palm beach" "swamp"))	*carol says to tom: the first section of the route goes between the palm beach and the swamp.*
tom makes a wh-ask(f) move with content not-ok	*tom says to carol: i do not understand.*
tom makes a wh-ask(i) move with content (describes "swamp" ?var205)	*tom says to carol: where is the swamp?*
carol makes a wh-ask(r) move with content (describes "swamp" ("between" "waterfall" "palm beach"))	*carol says to tom: the swamp is between the waterfall and the palm beach.*
tom makes a wh-ask(f) move with content ok	*tom says to carol: ok.*
tom makes a close(i) move with content (describes "first section" ("between" "palm beach" "swamp"))	*tom says to carol: i am done talking about the first section.*
carol makes a close(r) move with content ok	*carol says to tom: ok.*
tom makes a wh-ask(i) move with content (describes "second section" ?var215)	*tom says to carol: where is the second section?*
carol makes a wh-ask(r) move with content (describes "second section" ("between" "waterfall" "cliffs"))	*carol says to tom: the second section of the route goes between the waterfall and the cliffs.*
tom makes a wh-ask(f) move with content ok	*tom says to carol: ok.*

The full range of dialogues which can be run with the route giver having the top level goal can also be run with the route follower having the top level goal. These dialogues do not differ dramatically from the dialogues which we have already given, and so they are not duplicated here. The major difference is that new propositional content about sections of the route is requested by the route follower rather than offered by the route giver, leading to each section being initiated by a wh-question.

8.8 Dialogues JAM Can't Generate

Of course, there are many behaviours which occur in the human corpus that we can not generate. Most of these are due to not giving the agents a detailed enough representation of the maps; these deficiencies were discussed in section 5.1. However, there are also at least two dialogues which we would expect our agents to be able to have and which we still can not generate. This section gives these dialogues and explains what extensions would have to be made to the system in order to generate them. The changes to the JAM system which would

be required in order to generate the following dialogues are beyond the scope of the current research, and it is instructive that neither of the dialogues arise naturally from the way in which the system is built. Dialogues similar to the ones which we can not generate occur in the human corpus.

The first of the two dialogues is a variation of the dialogues given in section 8.6. In those dialogues, there are two relevant possible plan failures: the description of the first section of the route is the first, and the description of the swamp is the second. The difference between low and high risk plan commitment settings makes one agent replan as soon as the first failure of the dialogue occurs and the other repair at the first failure and then replan the description of the swamp at the second failure. In the dialogue which the current system can not generate, the agent repairs at the first failure and then replans the description of the first section when it encounters the second failure:

1A: The first section of the route goes between the palm beach and the swamp.
1B: I don't understand.
2A: Do you have the palm beach?
2B: Yes.
3A: Do you have the swamp?
3B: No.
4A: The swamp is between the waterfall and the palm beach.
4B: I don't understand.
5A: The first section of the route goes to the left of the palm beach.
5B: OK.

As we explained in section 7.3, an agent's choice to either repair or replan is governed by a threshold value which determines how much better than the current plan a replan must be in order to use it, and this threshold is lower for agents with high risk plan commitment settings than for agents with low risk ones. The plan commitment parameter also controls at which level the agent replans; a weighting system tends to make high level replans more desirable for agents with high risk plan commitment settings, and low level replans more desirable for agents with low risk plan commitment settings. Thus in section 8.6 anne, a high risk agent, replans the first section of the route at utterance 2A, whereas carol, a low risk agent, replans the explanation of the swamp at utterance 5A. It is carol's low risk plan commitment setting which makes her repair the first section of the route rather than replan it. Since her initial plan about the swamp (involving a yes-no question) fails, and her plan commitment setting makes her favour low level replans, she is unable at 5A to abandon her repair in favour of the higher level replan.

One way to avoid this problem while still allowing the agents to choose a strategy based on cost would be to create a new communicative posture parameter which allows the replanning

threshold and the weighting system for high versus low level replans to vary independently. However, this approach might be too general, since it might be possible that further study of the human corpus would show that agents decide between low and high level replans in some systematic way. In this case, more work could be done on the way in which the agents make cost estimates in order to match the human data.

The second of the two dialogues which we can not generate involves the amount of effort which the route giver uses when the first problem of the dialogue is encountered. We would expect our agents to be able to have the following dialogue:

> A: The first section of the route goes between the palm beach and the swamp.
> B: I don't understand. I don't have a swamp.

Here agent B's behaviour is somewhere between that of jack in section 8.2 and that of tom in section 8.3, since agent B doesn't just indicate that it doesn't understand but also doesn't create a plan to repair the failure. Of course, if we take the surface level behaviour "I don't have a swamp" to mean the same thing as tom's "where's the swamp?", then we can generate the structure of this dialogue. However, the same surface level of behaviour might mean "I didn't understand your utterance because I don't have the swamp, and I'm not proposing any particular way to recover from this failure"; this is the behaviour which we can not generate. One of the reasons is that the languages which the agents use for planning and communication (as described in sections 5.6 and 5.4 respectively) are not rich enough to express this information. Implementing the behaviour would also require a change to the way goal adoption is implemented. Currently agents either adopt the partner's goals entirely or not at all, but in this behaviour the agent must do some work towards the partner's goal (e.g., diagnosing the problem) but not all of it. We do not consider the necessary changes to be very difficult to make, but to do them properly would require more investigation of the human corpus to determine whether or not this behaviour does occur, and so we have left them to further work.

8.9 Conclusions

In this chapter, we have commented on a number of dialogues which the JAM system can produce as well as two which it can not but perhaps should. These dialogues encompass a whole range of behaviours associated with the human corpus, although, primarily because of limitations in the way the JAM dialogue agents represent knowledge about the maps, there are many human behaviours which can not be generated. Further work on the knowledge representation would allow more behaviours to be implemented, providing useful checks on the

analysis of the human corpus given in chapters 3 and 4.

Chapter 9

Conclusions and Further Work

The Principle of Parsimony states that by and large, agents try to complete tasks with as little effort as possible [Sha84]. This thesis demonstrates that this simple statement of Parsimony, taken as given, can be used to explain several different aspects of behaviour in human task-oriented dialogue by showing the effects of Parsimony in a corpus of human task-oriented dialogues about a map navigation task and by arguing that the analysis extends more generally to interactive, collaborative problem solving, and then by using the main points of the analysis to guide simulated conversations between two computer agents within the JAM system. Although this thesis does not present an argument for Parsimony in its own right or claim that the categories for which it argues must be psychologically plausible at any meaningful level, it does argue that the analysis is useful both as an explanation of human behaviour and for the purposes of building computer dialogue agents. It makes four major contributions: an analysis of "communicative posture", which is a range of choices in dialogue which can be characterised by decisions to be either careful or risky about different aspects of communication, an analysis of "recovery strategies" which allow the participants to recover from failures which have been brought about due to risky postures, a heuristic model of belief which risks failing to capture the full meaning of the dialogue in favour of efficiency, and a layered architecture which allows simulated dialogue agents to choose what actions to take based on the Principle of Parsimony.

Despite the fact that our four major contributions suffice for demonstrating the central thesis, there is still a great deal of interesting work which could be done in its support. This work divides into further work on empirical evidence, communicative postures, dialogue interaction, recovery strategies, the model of belief, and the layered architecture.

9.1 Further Work on Empirical Evidence

The map task works within a fairly limited and unnatural domain; although we are confident that the analysis of both the communicative postures and the recovery strategies will transfer to other domains, it would still be instructive to try them. However, even within the map domain there are a number of things which we could have changed about the task that would provide an interesting contrast to the existing data, both in terms of problems which we encountered with the experimental design and which may have affected the corpus and variations which would show up different aspects of communicative posture.

The way in which the task is set up allows an ambiguity of viewpoint for the task participants; most of the agents act as if they are pretending to walk along the route, but others act as if they are drawing the route on the map itself. The former makes analysis simpler because it restricts the kinds of referring expressions and descriptions that agents can construct; for instance, reusing part of an example from chapter 3, the drawing viewpoint allows descriptions such as:

> *MAP 10*
> L: my crashed plane is ABOVE + it's in the BASE of the quadrant + top right hand imaginary quadrant of the + erm + picture

There is a bewildering amount of choice available if the drawing viewpoint is taken, and since there is enough interesting behaviour which is produced by agents who use the role-playing viewpoint, it would be useful for task to be limited to it. However, some of the maps themselves are constructed in a way that favours the drawing viewpoint.[1] For instance, on map two, found in appendix A, the route very clearly swerves around an object which does not appear on the route giver's map but which does on the route follower's (the old well). Since the "explorer" drawing the route giver's map presumably would not have marked the route as going around an object but not have marked the object itself, this may lead agents to take the drawing viewpoint. This is especially the case where they do not discover that the partner has an object exactly where the route swerves (through a very high risk difference setting) before the route giver describes that section of the route, since, if the route giver has no object on which to base the description, he or she has little choice but to fall back on the drawing viewpoint. We believe that this flaw in the maps may introduce unnecessary confusion into the corpus. One way of forcing the agents into a role-playing viewpoint would be to vary the scale, origin, and orientation of the two maps, so that descriptions from the drawing viewpoint do not work.

[1] We assume that this is accidental, since the instructions for the task seem to favour the role-playing viewpoint.

Further work would be required to handle these changes in such a way that the cognitive load on the agents would not become too great.

There are also a number of changes to the experiment which would make an interesting contrast to the present analysis by way of testing individual communicative posture parameters. The specification and description resolution parameters could be more thoroughly tested by running experiments which make building referring expressions more difficult; for instance, the labels on the maps could be removed so that there is no default way of describing each of the map features, and more features which are semantically related could be included on the same map. The two focusing parameters could be investigated by looking at maps where the route is unexpected (such as in the map, given in appendix A, with east and west lake) and by including features with the same label but in totally different areas (such as in the castle map, which has two woods). Other changes would highlight other communicative posture parameters.

9.2 Further Work on Communicative Posture

It is obvious that it would be possible to improve our definition of communicative posture by carrying out a more complete analysis and to improve the JAM simulation by implementing more of the parameters and by extending the current implementations. However, the most pressing extension stems from the fact that in the JAM system, an agent's communicative posture is held constant for the duration of a dialogue. This is the most important way in which the simulation fails to match the dialogues in the corpus; human agents seem to be able to adapt their communicative postures based on what actions the partner has taken and how well they have fared in the dialogue so far. Thus JAM simulates limited segments of dialogue from the human corpus, but cannot simulate an agent's behaviour throughout an entire dialogue.

It seems likely that there are a number of relationships among the parameters settings which agents will wish to establish and maintain as they vary their communicative postures. The strongest of these occurs between an agent's difference and partner modelling parameters. It seems counter-intuitive for any agent to have a low risk difference setting, thus gathering information about the partner's knowledge, and a high risk partner modelling setting, which in effect throws away any information which is known. In addition, it seems likely that agents will move some pairs of parameters together simply because they help to provide the same level of detail in the dialogue. For instance, an agent might prefer to keep focus and context articulation at the same level, as well as focus and context resolution. These relationships among an agent's own parameters help make the agent appear consistent to the partner.

Besides maintaining relationships among their own parameter settings, agents might "balance" their settings against those of the partner for the sake of efficiency. For instance, if one agent has a low risk focus resolution setting, then there is no sense in the partner using low risk focus articulation, since the first agent will interrupt if it doesn't understand a focus shift. This kind of symmetry occurs for all of the parameters which occur in pairs, one for the explainer and one for the explainee. Finally, Simon Garrod (personal communication) is developing a theory of "focus systems" by which agents tend to mimic the decisions of their partners, using, for instance, the same kinds of referring expressions, complexity of descriptions, and so on. This helps make the agent more likely to be understood, since they know they are constructing utterances along the same lines as those of the partner. Much further work must be done in order to determine ways of varying the parameter settings which simulate behaviours in the human corpus and then to implement the variations in a working system using, for instance, game-theoretic techniques.

In addition to extending the treatment of communicative posture within dialogue, further work could be done on developing an analogous notion which applies to other types of human planning as well. Since, as we found in chapter 4, agents use recovery strategies during non-collaborative activities as well as during collaborative ones, it seems likely that we could devise a similar parameter system for non-collaborative planning domains. For instance, if an agent's plan to unlock a door fails, it might check a number of prerequisites of that plan: is it using the correct key? Is the key bent? Is the key inserted all the way into the lock? The fact that the agent may be able to repair the plan by discovering that any of these prerequisites is unfulfilled suggests that some high risk behaviour caused it not to check those prerequisites in the first place.

9.3 Further Work on Recovery Strategies

The recovery strategies used by the JAM system are only implemented very simply; these implementations could easily be improved. For instance, our implementation of goal adoption reconstructs only half of Allen's process for plan recognition 2.4.2. In addition, our **replan** operator is very inefficient in that it reconstructs from scratch a list of possible plans which should already be available from the **plan** operator in the form of a list of rejected plans. All of the recovery strategy implementations could be improved with further work; although this work is not necessary for proving the point of this thesis, it would be useful for future systems which were to use the recovery strategies.

A more pressing improvement of the recovery strategies involves widening the analysis and

associating it more closely with high risk communicative posture. There should be recovery strategies to handle all of the different ways in which high risk postures can introduce failures into the dialogue; we have identified some of the correspondences (for instance, that failures due to high risk difference are usually handled using repair, and that elaboration and omission work specifically for problems due to high risk ontology) but a more thorough analysis is in order. In addition, there are several suggestions for recovery strategies in the literature which are not handled in the present analysis. For instance, Cohen [Coh81] suggests that subdialogues which clarify referential miscommunication usually "downshift" into "low gear" by having the same form as the dialogues which they interrupt but a carefully thought out amount of detail. This strategy is related to both elaboration and repair, but isn't clearly either. Goodman [Goo86] presents a taxonomy of referential misunderstandings and some strategies which human agents use to recover from them; these strategies are primarily more specific forms of elaboration and omission. A wider analysis of the recovery strategies would match the high risk communicative postures better as well as relating to more of the strategies which have been suggested elsewhere.

9.4 Further Work on the Dialogue Interaction

JAM's top level control structure has one serious disadvantage: it can not be used for dialogues with more than two agents. We have shown in section 5.4 that for two-agent dialogues, the control fits the Sacks, Schegloff, and Jefferson [SSJ78] model of turn-taking; further work is needed to develop an arrangement for more agents which still gets the turn-taking correct. The main difficulty in devising a workable scheme will be that it will require some way of simulating simultaneous thought for the agents in order to give all non-speakers an equal chance to take over the turn at transition relevance points where the current speaker has not selected who is to speak next. Such a scheme would also benefit the two-agent dialogues because it would allow the initiative to be taken by whichever agent comes up with something to say first.

Our dialogue interaction would also be improved by using a version of dialogue games which is better attuned to the map task. Kowtko et al. [KID91] have developed a set of dialogue games which are specifically designed to match the structures which are present in the human map task corpus; she has found that there are much longer sequences of moves which tend to occur together than is suggested by the current two and three move games, and using her games would help make the simulated dialogues match behaviour in the corpus more closely over longer segments of text. In addition, the games which are available to the agents should be expanded so that agents can refuse to continue a game at any point. This could be done by allowing not just positive and negative moves, but also refusal moves which immediately

terminate the game. Refusal moves would allow the agents to interrupt each other, further improving the ability of agents to take the initiative from their partners.

Finally, further work on a flexible mapping between dialogue moves and surface level behaviours would provide better support for our claim that at the dialogue game level, we can best simulate the human dialogues by having agents generate all the moves which are necessary for completion of a game, even if at the surface level several moves are realised in one utterance or in non-linguistic behaviour such as nods and eye movement. Our claim that all of the dialogue moves exist in the structure no matter how they are realised is important to the belief model, since it requires all moves to be present in order to update beliefs correctly. Research which actually constructed a mapping would not only strengthen our claim that our simulated dialogues are similar to the corpus dialogues in the relevant ways, but would also shed light on important issues such as the role of coordination and non-linguistic behaviours in dialogue. Some important groundwork for this mapping will be done in the projects described in section 9.7, since both of these projects involve extending the JAM system so that it can produce natural language with a variety of syntactic constructions.

9.5 Further Work on the Model of Belief

There are two very obvious improvements which could be made to the model of belief. The first is to implement the more general interface described in chapter 6; this would not be difficult but it has not been done. The second is to implement the two-process model which is described rather than just the heuristic state transition process. We have sketched an algorithm for the interface between the two processes in section 6.6. Implementing the hybrid model would make it possible to experiment with under what circumstances the system should use each of the halves and would be a first step towards building a closer simulation of human belief modelling in a domain which has a deeper inference structure than the map task. The next step would be to build a compiler that could take the theorem proving half of the model and automatically create the heuristic mechanism, either before the beginning of a dialogue, or, more interestingly, as the dialogue progresses. Such a compiler could be motivated on a number of grounds. First of all, it would make the model more accessible for future dialogue systems, since compiling the model by hand is tedious. Second, it would investigate the difference between the heuristic mechanism as a computer program and the theorem prover as an executable specification, in software engineering terms. Third, it would provide an instance of a hybrid architecture in which the communication is different from the standard arrangements; usually a number of conscious rules control the use of a fast, subconscious process or a subconscious process control

the firing of a number of conscious rules, but in the proposed architecture, the subconscious mechanism would work wherever possible and only switch to the conscious process where necessary. Finally, it can be used as an analogy for how one learns to engage in dialogue in terms of human memory processes, and might be relevant to the discussion about the differences between competence and performance.

None of this further work would substantially change the basic mechanism behind the heuristic belief model. However, it would also be interesting to see how far one could push the analogy between the heuristic belief process and what happens in human memory. The heuristic mechanism is an agglomerative process, as Stenning [Ste91a] suggests human memory updating is, and it avoids the kind of variable rebinding that he claims is difficult for human memory [Ste91b]. It would be interesting to see if there are other variables which remain within the heuristic process which should also be "compiled out" or other changes to the belief process which will make it conform more closely to Stenning's theories about human memory updating; such work would be likely to be beneficial to work in memory as well as work in belief modelling, since it could be used to test some of the assumptions about computation in Stenning's theories. Another interesting extension would be to build a planner which reasons backwards from the goal state in the finite state mechanism (i.e., that state which most closely represents mutual belief) in order to decide what action to take next in the dialogue. Currently the plan operators and the finite state belief mechanism contain redundant information, and exploring how far the belief mechanism is reversible might point out interesting changes which could be made to it as well as developing a fast heuristic method of planning.

9.6 Further Work on the Layered Architecture

The layered architecture would benefit from being rewritten with a richer planning language so that the domain level operator schemas are easier to read and write. In addition, the communication between the layers could be improved by implementing a proper message passing system; for instance, the current broadcast facility requires a layer to have access to the names of all of the operators at the layer below, making the system more difficult to modify than it needs to be. Beyond this rational reconstruction, the layered architecture could be improved by making the planning operators more declarative. One motivation for building the layered architecture in the first place was that we noticed that the **repair** strategy as it is currently implemented can be represented in terms which are not very different from the representation of the schematic component of a domain level operator. One constraint for attempting repair is that the agent must be working on the goal to be repaired and that goal must have failed; in

our planning operator language, if the planning agent is ?A and the desired effect of the repair is the goal ?G, this can be represented as

```
(and (eq (g-node-goal (agent-dm ?A)) ?G)
     (eq (node-status (agent-dm ?A)) 'failed))
```

The other constraint is that one of the plan's prerequisites must have failed; if we imagine that the function "op-used" picks out the domain level operator which was used to plan the current goal (which can be read directly off the plan tree), the function "op-prereqs" picks out the prerequisites of an operator, and the function "failed?" tests whether or not a goal is satisfied, then we might construct a list of failed prerequisites as follows[2]:

```
(forall ?P in (op-prereqs (op-used (agent-dm ?A)))
        (if (and (typep ?P 'goal)
                 (failed? ?P))
            ?P)))
```

Thus the operations list of the repair is just this list of failed prerequisites, and the second constraint is that this list not be empty. We think that it is probably possible to rewrite at least the strategy level operators to match the language of the domain level schemas, and that such a move would increase the elegance of the system as well as providing an interesting relationship to, for instance, past work on Prolog meta-interpreters.

Finally, in addition to changes to the style of layering, the architecture could be improved by exploiting a natural division of the domain level operators into subsets which correspond to the different types of planning used to group the communicative posture parameters in chapter 3. By and large when an agent builds a plan for a section of the route, it works through the groups of operators in order, first planning the task, second setting up the discourse structure, and finally realising the utterance. Thus it would make sense for the planner to decide which group of operators to consider first based on the current context rather than doing a brute force search as in the current system. Of course, the difference would become more important in a version of the system which had more than the seven domain level operators used by the current system. In a large system, control of the domain level operators might be an interesting problem in its own right; for instance, work by Gerlach and Horacek [GH89] describes a way of grouping inference rules (or plans in which the action may be empty) which describe dialogue so that they can be applied more efficiently.

[2] Just as with our current implementation of repair, this definition only works if goals add up monotonically, so that a fulfilled goal never becomes unfulfilled later on in the execution of the plan.

9.7 Other Further Work

In addition to the further work described above which supports the core ideas of the thesis, the JAM simulation is useful as support for research in other areas, since it generates a variety of dialogues with different structures from which specific aspects of dialogue can be explored. In this section, we describe two projects which will extend the JAM system in an investigation of other research areas.

The first is an ESRC supported project entitled "Pragmatic Factors in the Intonation of Synthesised Spoken Dialogue"; it is led by Bob Ladd at the Linguistics Department of Edinburgh University. Its goal is to build a computational model of the interaction between pragmatic factors of dialogue (such as shared knowledge, discourse structure, and speech act types) and features of intonation (such as pitch accent, phrase accent, boundary tones, and pitch range). Part of the research will involve extending the JAM system by adding a simple natural language generator and a model of intonation which takes into account pragmatic factors.

The second is a Joint Councils Initiative supported project entitled "Natural Language Generation under Time Constraints"; it is led by Stephen Isard at the Human Communication Research Centre of Edinburgh University. Its goal is to characterise human behaviour during language production under time constraints which do not allow fully grammatical utterances to be produced. Part of the research will involve adding a natural language generator to the JAM system which, when pressed for time, will produce the same kinds of false starts and self-corrections that occur in the human map task dialogues.

9.8 Conclusions

This thesis demonstrates that a naive version of the Principle of Parsimony can explain, at least at a superficial level, several different aspects of human behaviour during task-oriented dialogue, and that making a computer dialogue agent act parsimoniously results in dialogues with more natural structures. It divides into two parts: an analysis of a human corpus of dialogues about a map navigation task, which demonstrates how Parsimony affects the structure of the dialogues, and the development of a computer system, JAM, which demonstrates that the main points of the analysis can be incorporated into a working simulation of the human data. This thesis has made four major contributions: two concerning the analysis of the corpus and two concerning the development of the JAM system.

The first major contribution is an analysis of a number of aspects of task-oriented dialogue

along which agents must make decisions which can be characterised by "risk"; agents can either risk successful completion of the current goal on the current attempt, taking the chance that recovery from a failure will be needed, or expend extra effort in order to make the current attempt more likely to succeed. According to the Principle of Parsimony, agents will make whatever choices they believe will lead to completion of the task with the least amount of effort overall. We identify twelve different aspects of the dialogue for which choice is governed by Parsimony, and, following Shadbolt [Sha84], we term them "communicative posture parameters". Although most of our examples come from a map navigation domain, we argue that our analysis applies more generally to interactive, collaborative problem-solving.

The second major contribution is an analysis of recovery strategies which agents can use in order to recover from plan failures that have been caused by choosing high risk communicative posture settings. We identify nine recovery strategies in all, some of which are actions which an agent can take in order to recovery directly from a failure and others of which only contribute indirectly to the solution of the problem. In addition, some of the recovery strategies apply generally in all planning domains, while others work specifically in collaborative domains where communication is necessary. The first and second contributions together demonstrate how the Principle of Parsimony affects the style of the dialogue interaction, and especially shows that one of the things which makes human dialogue seem more natural than that generated by computer systems is that the human agents tend to take quite substantial risks and rely on the robust nature of dialogue in order to recover from any subsequent failures.

The third major contribution is a model of belief which risks failing to capture the meaning of a dialogue fully by choosing to operate more efficiently but less precisely than previous models of belief. Previous models have used theorem provers on knowledge bases of beliefs and typical inference rules in order to derive all beliefs which are logical conclusions of the base set of beliefs. They have the disadvantage that they are too powerful as a model of what humans ordinarily conclude; humans can sometimes work out all of the logical conclusions of a set of beliefs with a vast amount of effort by using theorem proving, but they more often reason incompletely and use much less effort. Some techniques have been developed for limiting the power of the logical approaches, but at the expense of the elegance of the approach. Our own model of belief operates heuristically by failing to make some kinds of inferences in order to simulate the results of ordinary human belief modelling better than the previous models. We also describe how it could be integrated with a theorem proving approach in order to allow the agent to use the heuristic approach whenever possible but do more complete reasoning when that is required.

The fourth major contribution is a system architecture which allows agents to always choose

what actions to take in the dialogue based on the amount of effort which remains in order to complete the entire task. This architecture treats all possible ways of continuing from a point in the dialogue uniformly and includes a representative sample of the recovery strategies from the analysis: planning, communicating, adopting the partner's goals, replanning, and repair. Previous systems have only implemented one recovery strategy at a time or have had very rigid ways of choosing which one of the recovery strategies to attempt. The JAM architecture allows the choice to be made more flexibly. In addition, the construction of the system allows changes to the way in which the agent decides upon its actions to be made easily, making it simple to vary the simulation and investigate the effects of different dialogue strategies. The JAM architecture is useful not just for simulating task-oriented dialogue but also more generally for systems in which there are a number of different ways to continue from each decision point and there is a principled way in which to make the choice. The third and fourth contributions combine to demonstrate that Parsimony can be used in a working system in order to more accurately simulate the human dialogues.

These four contributions demonstrate the central thesis that many decisions in human task-oriented dialogues can be characterised in terms of minimising the total amount of effort spent on completing the task, and that if a computer system is constructed which also minimises effort, the resulting dialogue will simulate the human dialogues more closely as well as seem more natural than dialogue generated by systems which do not take effort into account.

Appendix A

The Maps

This appendix contains the instructions for the map task and the maps which accompany the example dialogues given in the thesis, with the route giver's map on the left and the route follower's map on the right. Where we give examples from only one set of agents for a pair of maps, we show the route which the route follower has drawn. For maps 9 and 10, which were from the early trials used by Shadbolt, there were several pairs of agents using the same maps and we do not give the routes which were drawn.

INSTRUCTIONS FOR THE MAP TASK

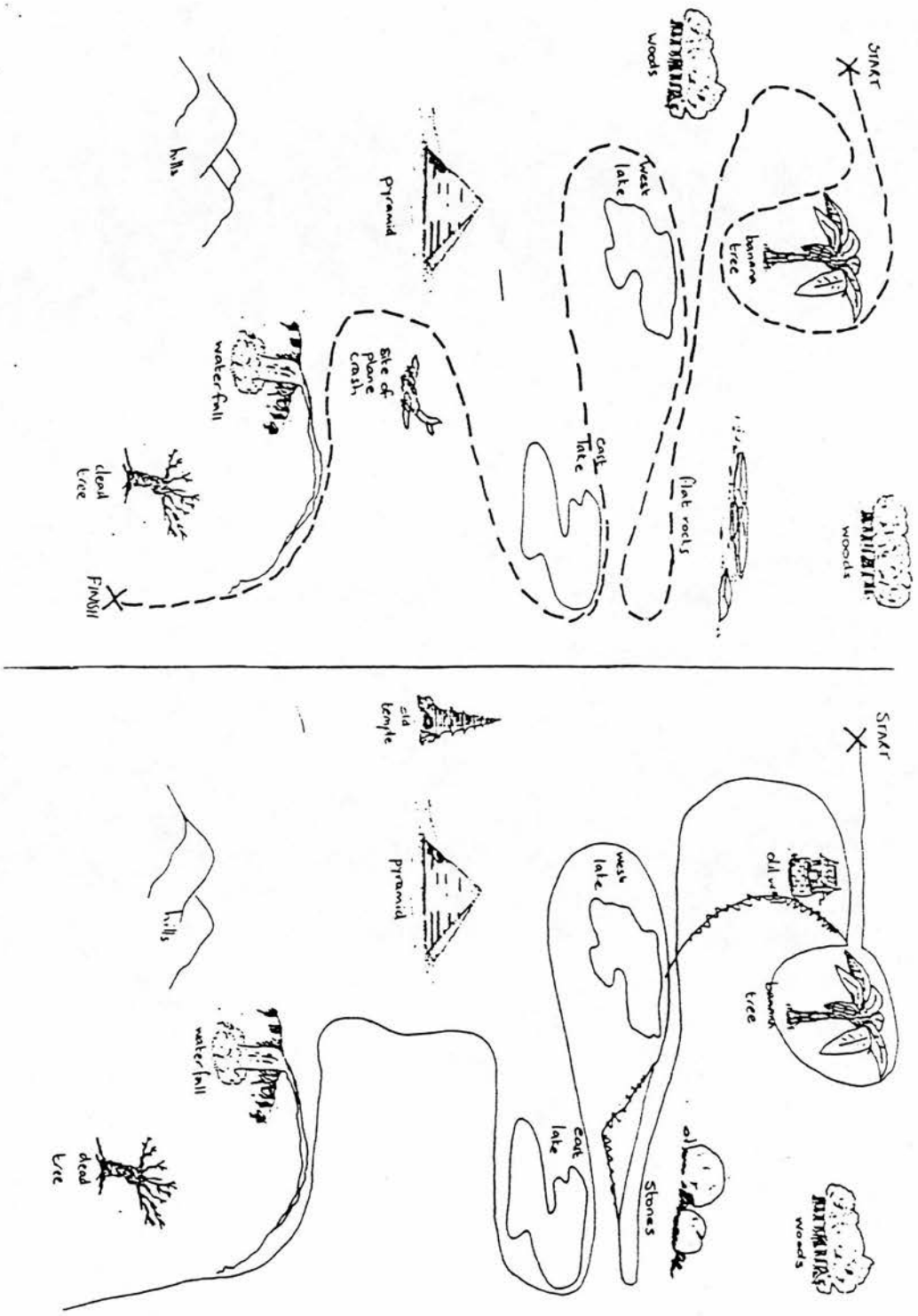
INSTRUCTIONS GIVEN TO THE ROUTE DESCRIBER (A)

YOU AND YOUR PARTNER HAVE BOTH GOT A MAP OF THE SAME PLACE. YOU'VE GOT A ROUTE MARKED ON YOUR MAP. ITS THE ONLY SAFE ROUTE THROUGH ALL THE DANGERS. YOUR PARTNER HASN'T GOT A ROUTE MARKED ON THEIR MAP. YOUR JOB IS TO DESCRIBE THE ROUTE TO YOUR PARTNER SO THAT HE/SHE CAN MARK IT ON THEIR MAP. YOU HAVE TO DESCRIBE IT EXACTLY BECAUSE ITS THE ONLY SAFE ROUTE. THE MAPS HAVE BEEN DRAWN BY DIFFERENT EXPLORERS, SO THEY MIGHT NOT BE BOTH THE SAME. THERE COULD BE SOME DIFFERENCES.

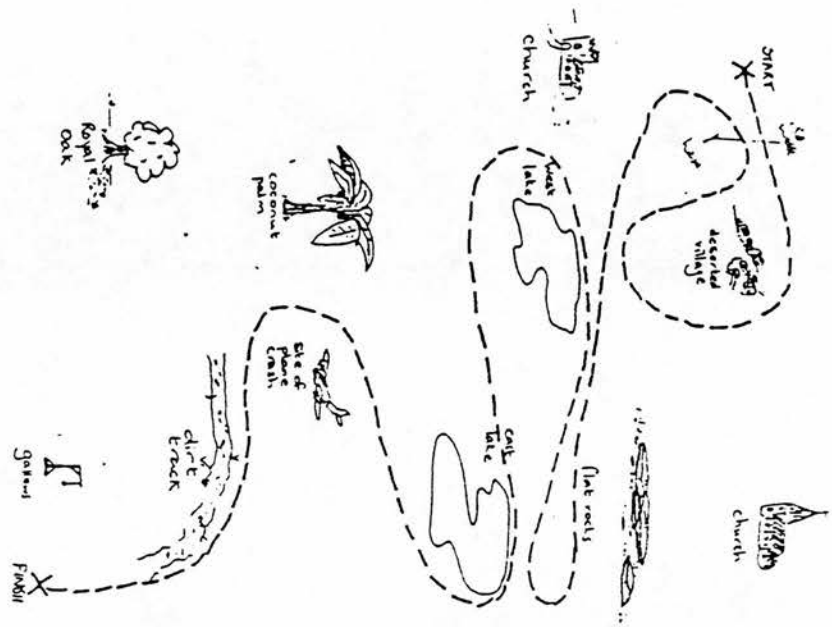
INSTRUCTIONS GIVEN TO THE ROUTE DRAWER (B)

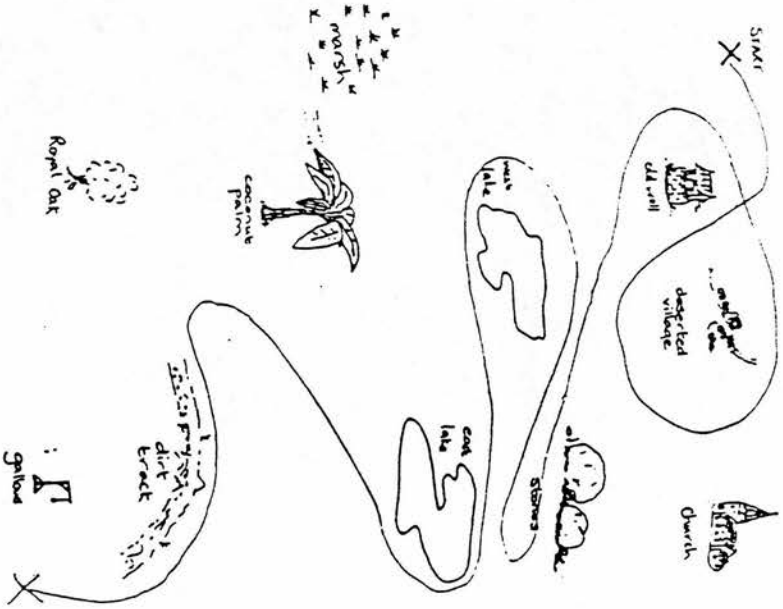
YOU AND YOUR PARTNER HAVE BOTH GOT A MAP OF THE SAME PLACE. HE/SHE HAS GOT A ROUTE MARKED ON HIS/HER MAP YOUR JOB IS TO FINISH UP WITH THAT ROUTE MARKED ON YOUR MAP. LISTEN TO WHAT HE/SHE SAYS AND ASK HIM/HER ANY QUESTIONS YOU WANT. YOU MUST DRAW IT EXACTLY. ITS THE ONLY SAFE ROUTE. THE MAPS HAVE BEEN DRAWN BY DIFFERENT EXPLORERS SO THEY MIGHT NOT BE THE SAME, THERE COULD BE SOME DIFFERENCES.

MAP 1

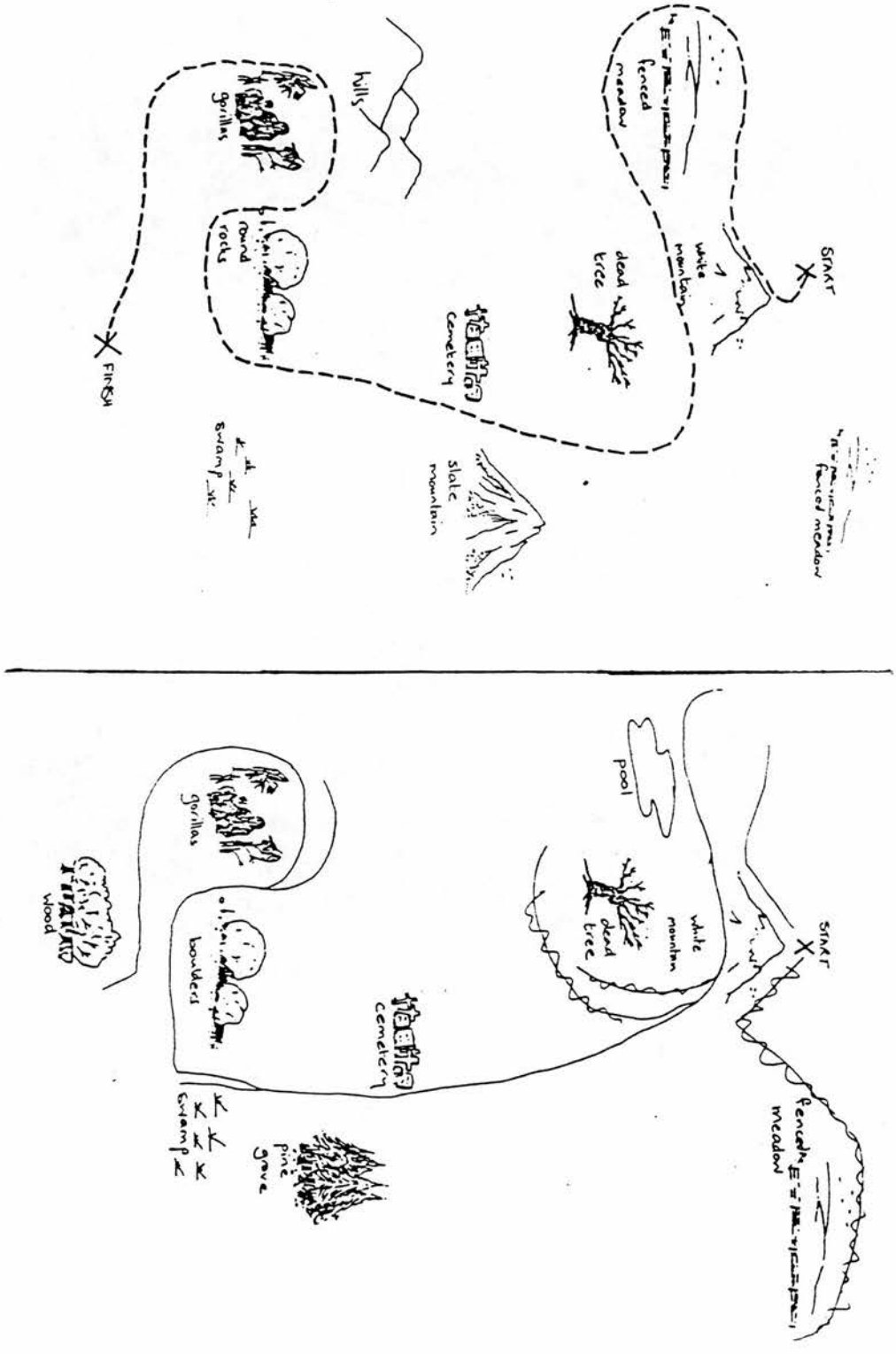

START
banana tree
woods
west lake
flat rocks
east lake
pyramid
hills
site of plane crash
waterfall
dead tree
FINISH
old temple
old well
stones

MAP 2

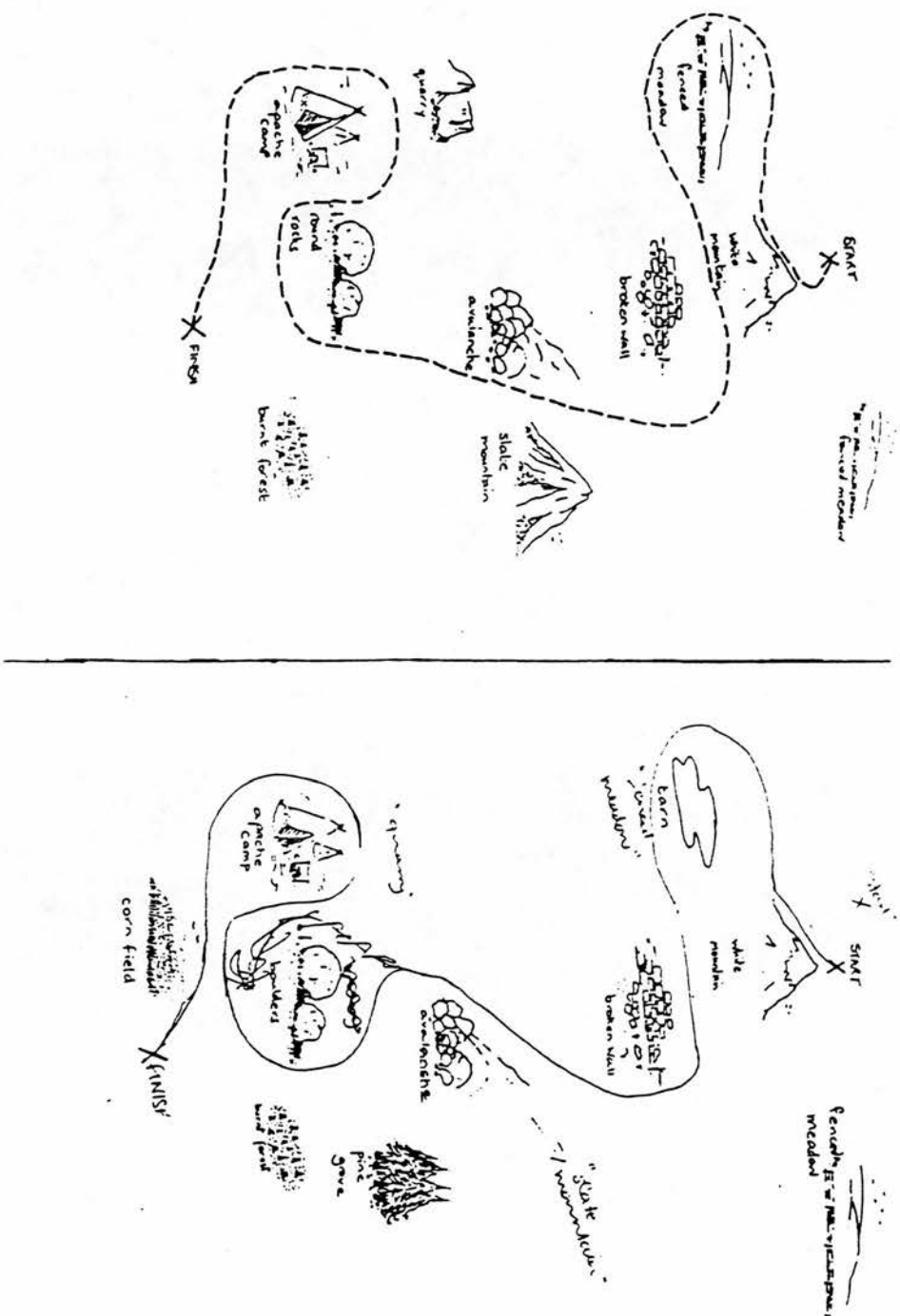

START
church
deserted village
west lake
flat rocks
east lake
dirt track
FINISH
Royal Oak
coconut palm
church


START
deserted village
west lake
east lake
stones
church
dirt track
FINISH
Royal Oak
coconut palm
marsh

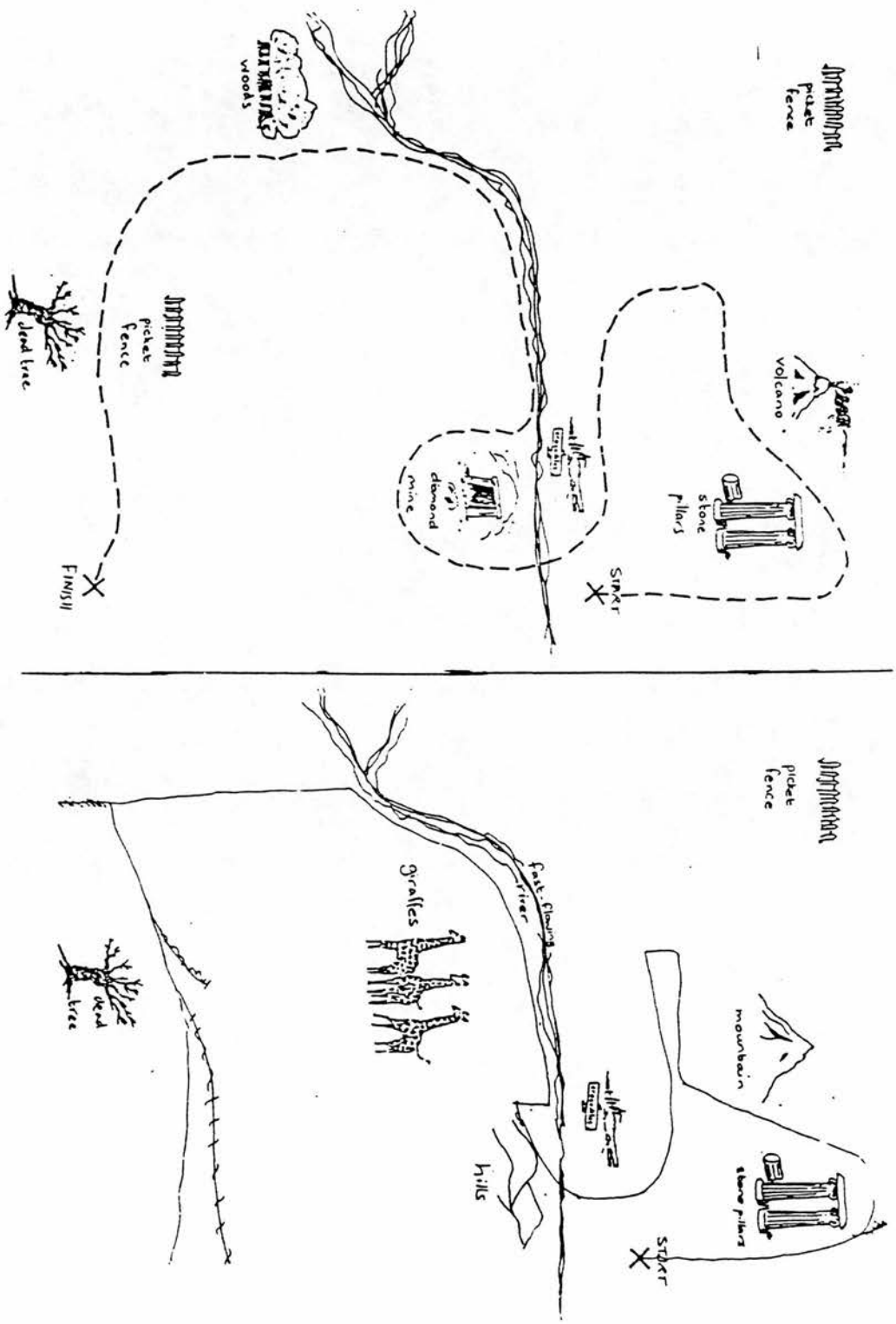
MAP 3


START
fenced meadow
white mountain
dead tree
cemetery
slate mountain
swamp
round rocks
gorillas
hills
FINISH
pool
boulders
pine grove
wood

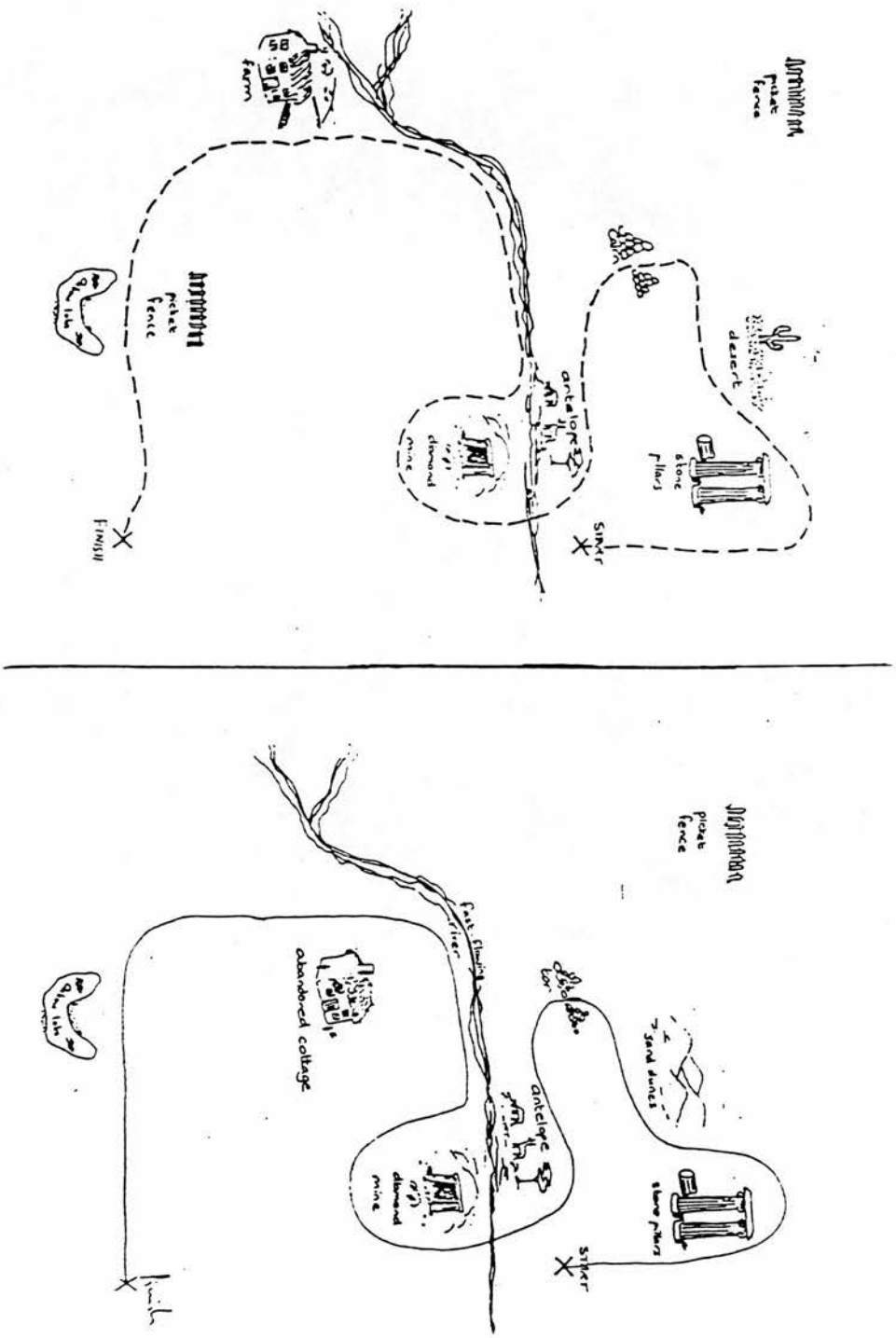
MAP 4


START
white mountain
meadow
broken wall
avalanche
slate mountain
burnt forest
FINISH
apache camp
quarry
round rocks
START
white mountain
meadow
broken wall
avalanche
pine grove
boulders
apache camp
corn field
FINISH

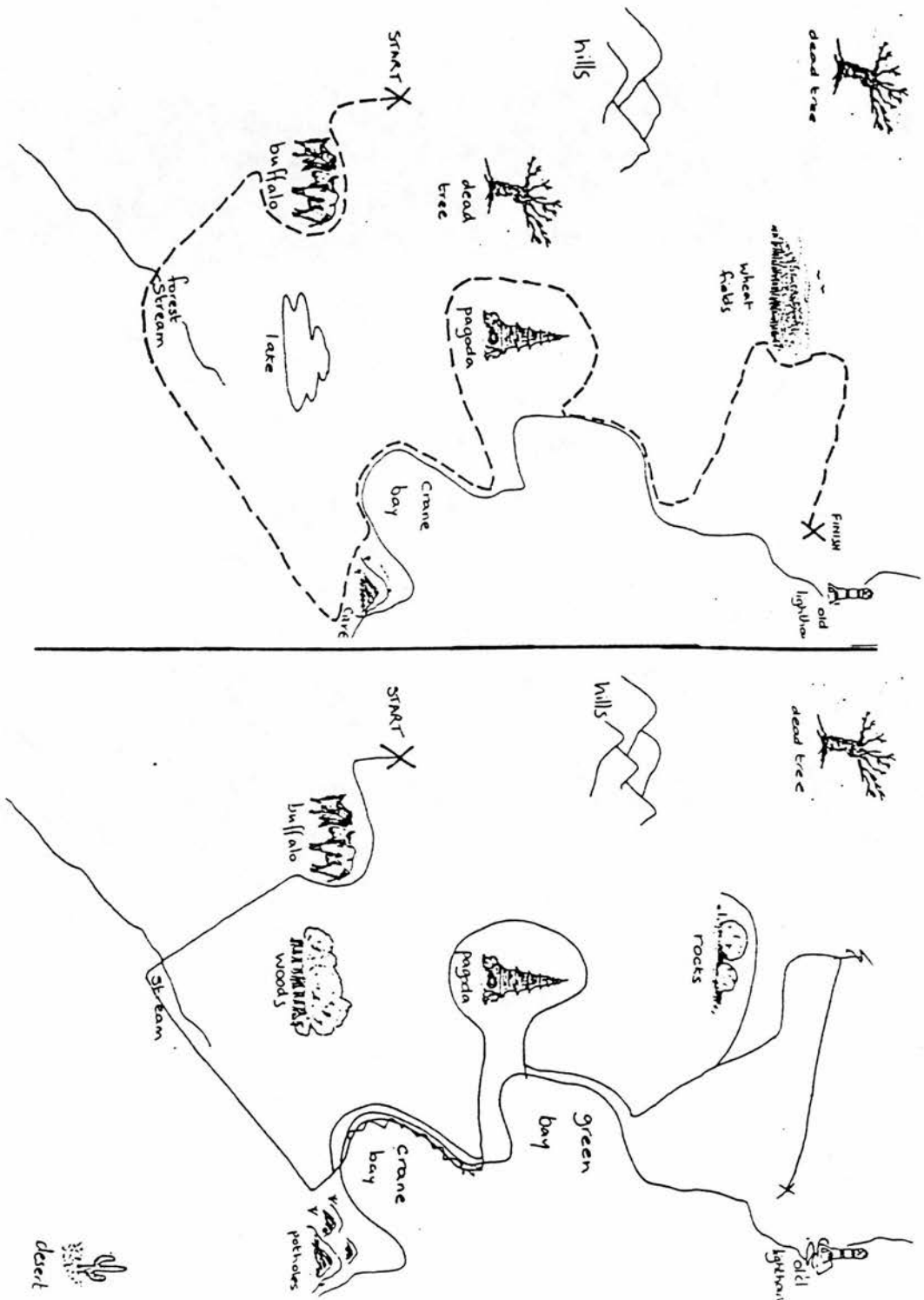
MAP 5


picket fence
woods
dead tree
picket fence
FINISH
diamond mine
volcano
stone pillars
START
picket fence
giraffes
dead tree
hills
mountain
stone pillars
START

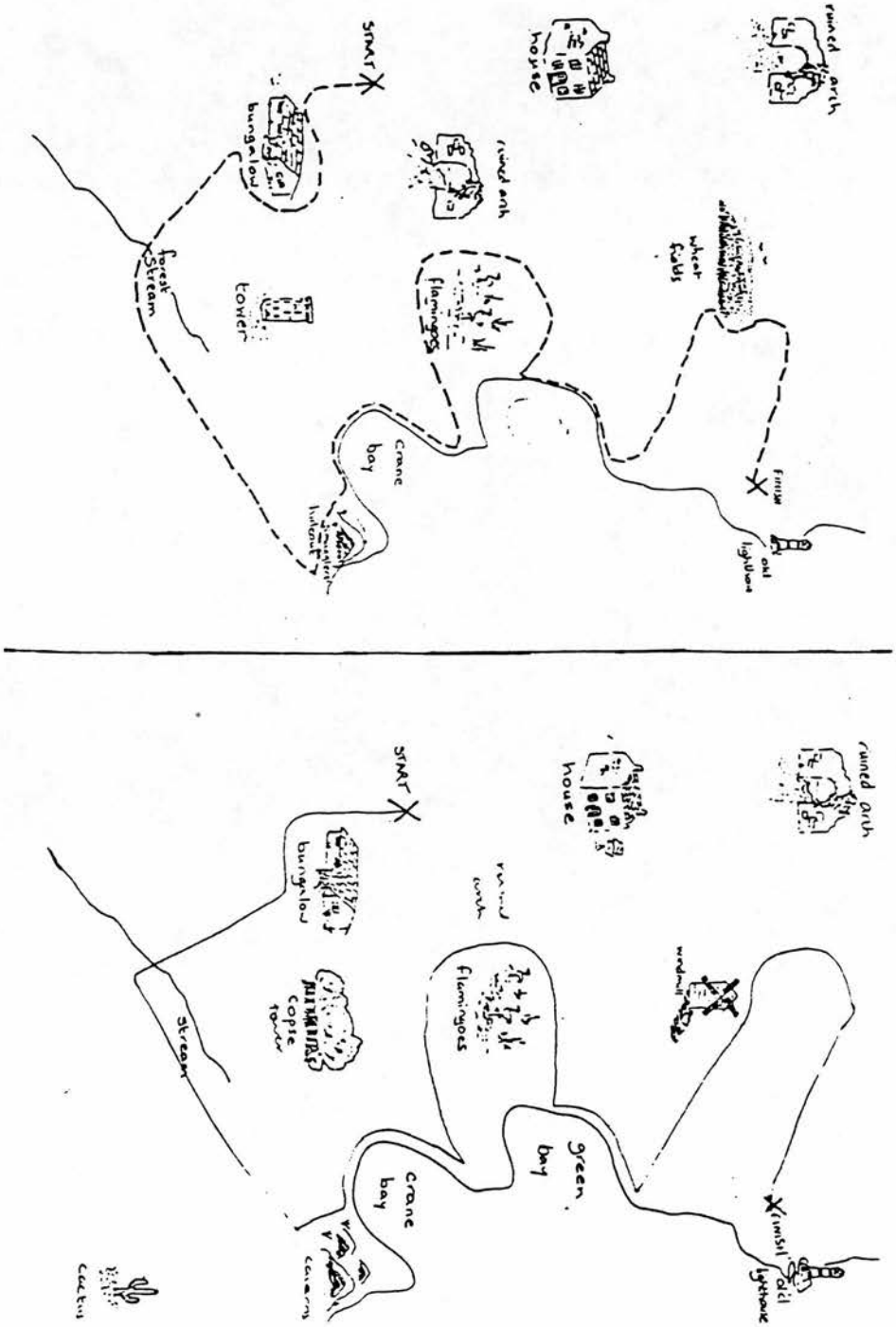
MAP 6


FINISH
SINKER
desert
stone pillars
antelope
diamond mine
abandoned cottage
sand dunes
picket fence

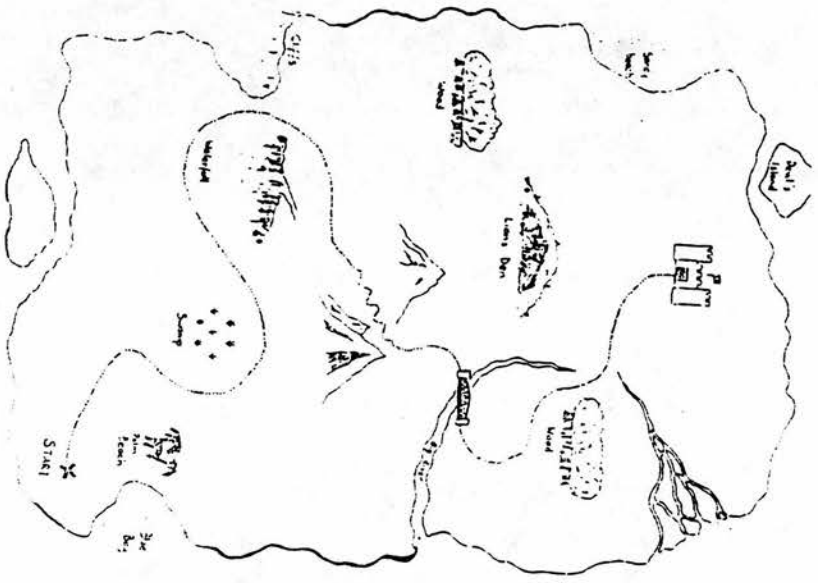
MAP 7

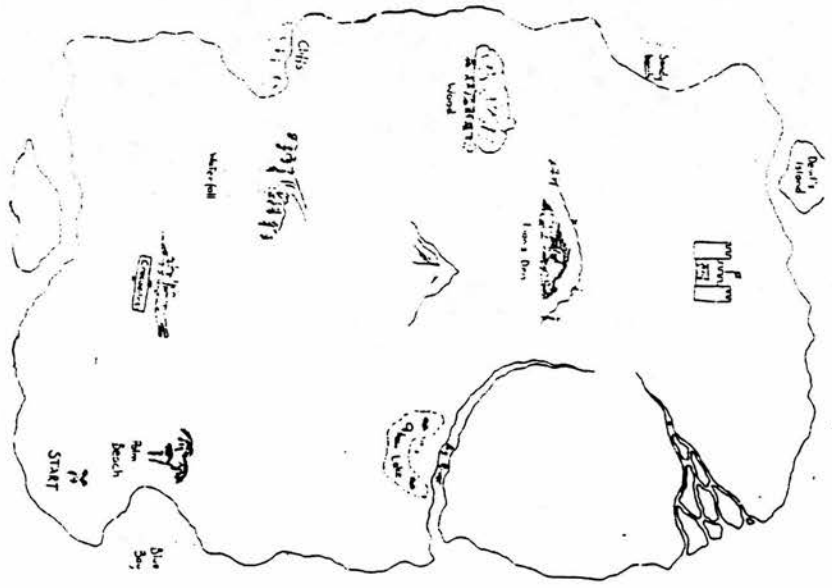

dead tree
hills
START
buffalo
dead tree
wheat fields
forest stream
lake
pagoda
crane bay
cave
FINISH
old lighthouse
dead tree
hills
START
buffalo
rocks
woods
pagoda
green bay
stream
crane bay
potholes
old lighthouse
desert

MAP 8

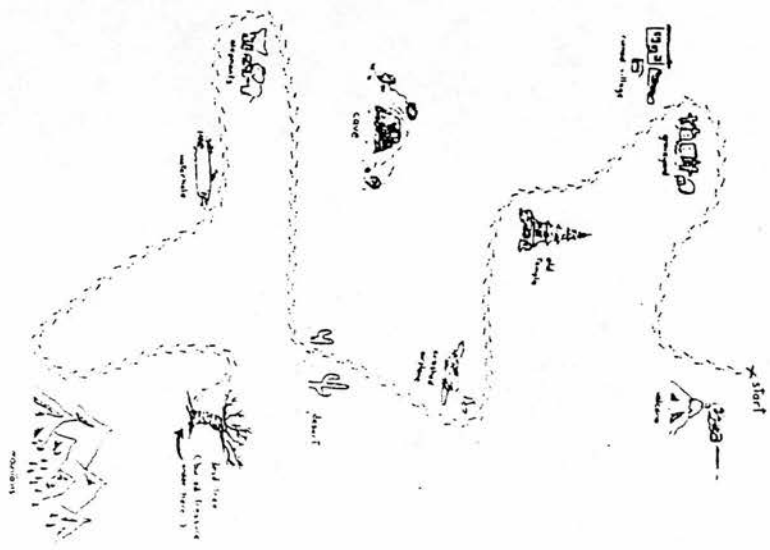

ruined arch
house
wheat fields
old lighthouse
finish
crane bay
flamingoes
forest stream
tower
bungalow
start
green bay
cope tower

MAP 9

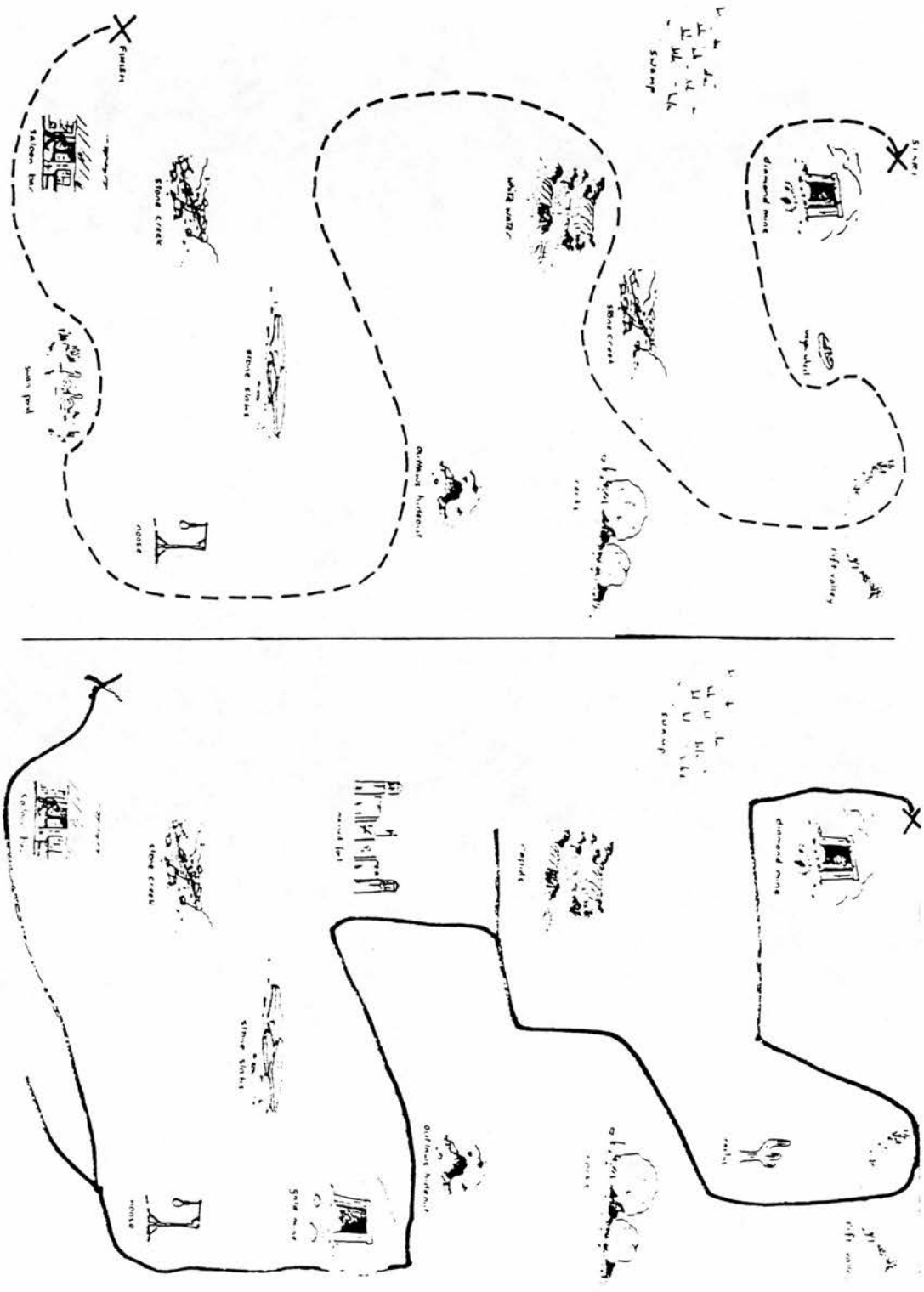




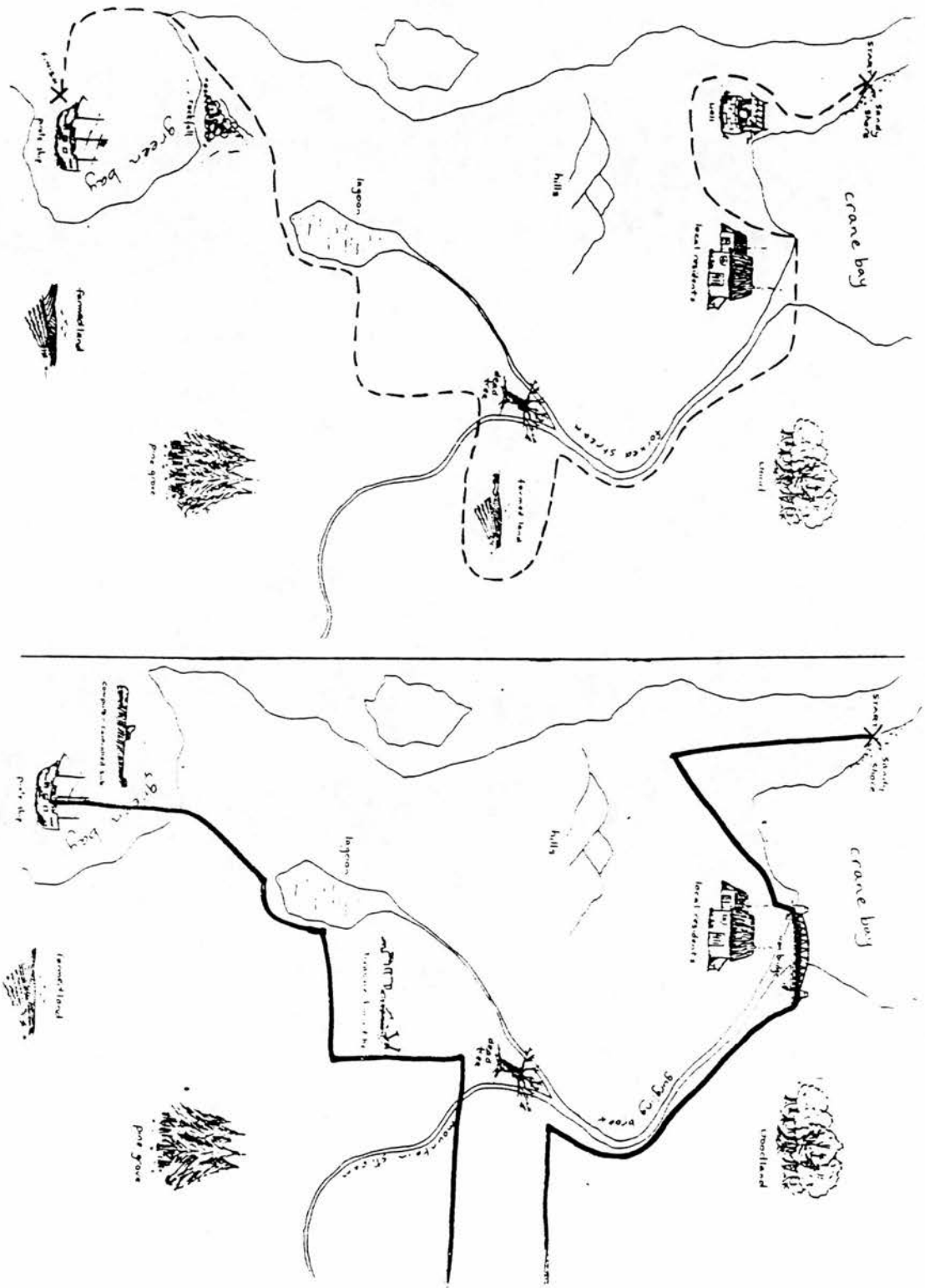
MAP 10



MAP 11



MAP 12


crane bay
hills
lagoon
pine grove
crane bay
hills
lagoon
pine grove

Appendix B

Space Calculations

Let a be the number of agents, g be the number of degrees of acquaintance, and n be the number of nesting levels. Suppose that every degree of acquaintance is accessible to every agent.[1] Furthermore, suppose that the representing agent does not wish to represent any agent's view of its own beliefs.[2] Let $S_C(n)$ be a recurrence relation on n which specifies the total number of belief states that will be needed to represent certain knowledge up to n levels of nesting. Then

$$S_C(0) = g,$$

since g is the number of degrees of acquaintance which any agent can hold with respect to the predicate or object. Furthermore, for all $n > 0$, $S_C(n)$ is the number of states needed to represent dialogue up to $n-1$ levels of nesting times the number of different positions the agent can take for the nth level of nesting. Any beliefs at the nth level of nesting must have the form

α_0 thinks α_1 thinks ... thinks α_n has γ acquaintance with the predicate or object,

where there are g choices for γ. Since we are representing the views of a particular agent, α_0 must be that agent. Moreover, for all $i > 0$, α_i must not be the same agent as α_{i-1}. Therefore,

[1] This assumption just means that there are no degrees like "unaware" which agents cannot ascribe to themselves or agents who are in some way defective, so that they can hold any of the degrees of acquaintance with respect to any predicate or object.

[2] For example, we don't want to represent what A thinks of A's degree of acquaintance or what A thinks B thinks of B's degree of acquaintance. We assume that, for the purposes of most domains, an axiomatisation of belief which includes something like Believes(A,Believes(A,P)) $\leftrightarrow$ Believes(A,P) is appropriate. A similar argument can be constructed without this constraint.

working from left to right, there are $a-1$ choices for each of the n remaining slots: a choices minus the one which was used in the slot before. Therefore, for all $n > 0$,

$$S_C(n) = S_C(n-1)g(a-1)^n.$$

Eliminating the recurrence,

$$S_C = g^{n+1}(a-1)^{\frac{1}{2}n(n+1)}.$$

Similarly, let S_E be the number of states needed to represent only the certain beliefs plus one value for uncertainty at every nesting level. Then adding uncertainty is just like adding one more degree of acquaintance for all but the zeroeth level of nesting, and

$$S_E = g(g+1)^n(a-1)^{\frac{1}{2}n(n+1)}.$$

The calculation of S_U is somewhat more complicated. Let $S_U(n)$ be a recurrence relation on n specifying the number of states necessary to distinguish among all of the different kinds of uncertainty which can occur up to n levels of nesting. Then $S_U(0) = S_C(0) = g$, since one cannot be uncertain about one's own degree of acquaintance. Moreover, using an argument similar to that for the calculation of S_C,

$$S_U(n) = S_U(n-1)(a-1)^n U(n)$$

for all $n > 0$, where $U(n)$ is the number of different positions (including uncertain ones) the agent might take with regard to a belief at the nth level of nesting. Consider $U(1)$. Let γ be the set of certain positions the agent might take (i.e., the set of degrees of acquaintance); then $\mid \gamma \mid = g$. But the agent might be unable to choose between these degrees of acquaintance in any possible way except that the agent must venture some opinion, even if that opinion is that any one of the positions might hold. Therefore the set of all possible positions, certain and uncertain, is the power set of *gamma* minus the empty set, and so the number of possible positions is $2^g - 1$. Therefore $U(1) = 2^g - 1$. But at any level of nesting beyond the first one, the set of all possible positions is the power set of the positions at the level before minus the empty set. Therefore for all $n > 1$,

$$U(n) = 2^{U(n-1)} - 1.$$

Recapitulating,

$$S_U(0) = g,$$

and for all $n > 0$,

$$S_U(n) = S_U(n-1)(a-1)^n U(n),$$

where $U(n)$ is the number of different positions the agent might take with regard to a belief at the nth level of nesting.

$$U(1) = 2^g - 1,$$

and for all $n > 1$,

$$U(n) = 2^{U(n-1)} - 1.$$

Simplifying in the same way as for S_C, when $n = 0$, $S_U = g$, and for $n > 0$,

$$S_U = g(a-1)^{\frac{1}{2}n(n+1)} \prod_{i=1}^{n} U(n),$$

where $U(1) = 2^g - 1$ and for $n > 1$,

$$U(n) = 2^{U(n-1)} - 1.$$

Appendix C

The Belief Model for the Simulation

This appendix gives the state transition network in tabular form for the implemented eighteen state belief model described in section 6.7. The states are numbered as follows:

1. The self is aware of the concept, is uncertain about the partner's position with regard to the concept, and is uncertain about what the partner thinks the self's position is with regard to the concept.
2. The self is aware of the concept, thinks the partner is aware of the concept, and is uncertain about what the partner thinks the self's position is with regard to the concept.
3. The self is aware of the concept, thinks the partner holds the concept vivid, and is uncertain about what the partner thinks the self's position is with regard to the concept.
4. The self holds the concept vivid, is uncertain about the partner's position with regard to the concept, and is uncertain about what the partner thinks the self's position is with regard to the concept.
5. The self holds the concept vivid, thinks the partner is aware of the concept, and is uncertain about what the partner thinks the self's position is with regard to the concept.
6. The self holds the concept vivid, thinks the partner holds the concept vivid, and is uncertain about what the partner thinks the self's position is with regard to the concept.
7. The self is aware of the concept, is uncertain about the partner's position with regard to the concept, and thinks that the partner thinks that the self is aware of the concept.
8. The self is aware of the concept, thinks that the partner is aware of the concept, and thinks that the partner thinks that the self is aware of the concept.

9. The self is aware of the concept, thinks that the partner holds the concept vivid, and thinks that the partner thinks that the self is aware of the concept.
10. The self holds the concept vivid, is uncertain about the partner's position with regard to the concept, and thinks that the partner thinks that the self is aware of the concept.
11. The self holds the concept vivid, thinks that the partner is aware of the concept, and thinks that the partner thinks that the self is aware of the concept.
12. The self holds the concept vivid, thinks that the partner holds the concept vivid, and thinks that the partner thinks that the self is aware of the concept.
13. The self is aware of the concept, is uncertain about the partner's position with regard to the concept, and thinks that the partner thinks that the self holds the concept vivid.
14. The self is aware of the concept, thinks that the partner is aware of the concept, and thinks that the partner thinks that the self holds the concept vivid.
15. The self is aware of the concept, thinks that the partner holds the concept vivid, and thinks that the partner thinks that the self holds the concept vivid.
16. The self holds the concept vivid, is uncertain about the partner's position with regard to the concept, and thinks that the partner thinks that the self holds the concept vivid.
17. The self holds the concept vivid, thinks that the partner is aware of the concept, and thinks that the partner thinks that the self holds the concept vivid.
18. The self holds the concept vivid, thinks that the partner holds the concept vivid, and thinks that the partner thinks that the self holds the concept vivid.

This state numbering system is summarised in figure C.1.

Figure C.1: The States for JAM Agents

Σ thinks Π is	Σ is aware	Σ is vivid	Σ thinks Π thinks Σ is
uncertain	1	4	uncertain
aware	2	5	uncertain
vivid	3	6	uncertain
uncertain	7	10	aware
aware	8	11	aware
vivid	9	12	aware
uncertain	13	16	vivid
aware	14	17	vivid
vivid	15	18	vivid

Figure C.2: The State Transition Table for JAM Agents

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\Sigma : W_I$	7	⊥	9	⊥	⊥	⊥	7	⊥	9	⊥	⊥	⊥	7	⊥	9	⊥	⊥	⊥
$\Sigma : W_{RP}$	⊥	⊥	⊥	⊥	16	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	16	⊥
$\Sigma : W_{RN}$	⊥	8	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	8	⊥	⊥	⊥	⊥
$\Sigma : W_{FP}$	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : W_{FN}$	⊥	⊥	9	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : I_I$	⊥	⊥	⊥	16	16	⊥	⊥	⊥	⊥	16	16	⊥	⊥	⊥	⊥	16	16	⊥
$\Sigma : I_{RP}$	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : I_{RN}$	⊥	⊥	9	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : Y_I$	1	1	1	4	4	4	7	7	7	10	10	10	13	13	13	16	16	16
$\Sigma : Y_{RP}$	⊥	⊥	⊥	16	17	18	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : Y_{RN}$	7	8	9	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : O_I$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\Sigma : O_{RP}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\Sigma : O_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Sigma : C_I$	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥	⊥	⊥	⊥	18
$\Sigma : C_{RP}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	18
$\Sigma : C_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	9	⊥	⊥	⊥
$\Pi : W_I$	2	2	2	5	5	5	2	2	2	5	5	5	14	14	14	17	17	17
$\Pi : W_{RP}$	⊥	⊥	⊥	⊥	⊥	⊥	3v6	⊥	3v6	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Pi : W_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	8	⊥	8	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Pi : W_{FP}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥
$\Pi : W_{FN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	17	⊥	⊥
$\Pi : I_I$	3v6	3v6	3v6	6	6	6	3v6	3v6	3v6	6	6	6	3v6	3v6	3v6	6	6	6
$\Pi : I_{RP}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	18	⊥	⊥
$\Pi : I_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	17	⊥	⊥
$\Pi : Y_I$	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
$\Pi : Y_{RP}$	3	⊥	⊥	6	⊥	⊥	9	⊥	⊥	12	⊥	⊥	15	⊥	⊥	18	⊥	⊥
$\Pi : Y_{RN}$	2	⊥	⊥	5	⊥	⊥	8	⊥	⊥	11	⊥	⊥	14	⊥	⊥	17	⊥	⊥
$\Pi : O_I$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\Pi : O_{RP}$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\Pi : O_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥
$\Pi : C_I$	15	15	15	18	18	18	15	15	15	18	18	18	15	15	15	18	18	18
$\Pi : C_{RP}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	18
$\Pi : C_{RN}$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	17

Given these meanings for the state numbers, figure C.2 gives the state transitions used by the agents of the JAM system. The left hand column gives the move and the column headings give the initial state. Where the entry in the table is a number, then the state transition takes the agent to that state. If the entry in the table is u(3), then the move is being made by the partner and involves new propositional content. The state transition takes the agent to state 3 if the agent does not have vivid all of the concepts which combine to form the proposition and to state 6 otherwise.

Appendix D

How the Belief Model is Implemented

The belief model is used in two ways, as described in section 6.2: it handles updating the beliefs of the agent based on utterances that the agent has made or heard, and it allows the agent to query it about its own beliefs.

D.1 How Belief Updating is Implemented

Calculating the belief state update for the central concept of a dialogue game involves determining the old state of the concept, looking up the corresponding transition in the state transition table, and storing the new state for the concept. This is implemented in an inefficient but very simple-minded way. Concepts are data structures which contain names, states, a list of referring expressions, the type of node which they are in the concept network (AND, OR, or PRIMITIVE), a list of pointers to parent concepts, and a list of pointers to child concepts. This network of concepts can be indexed in two ways: a hash table indexes the concepts by their names, and a list of pointers to each concept allows the network to be searched by referring expression. Of course, it would be more efficient to write the referring expression index as a second hash table or as a trie (a tree in which every node has twenty-six ordered children and in which entries keyed on alphabetic identifiers can be found by reading the identifiers vertically down the branches [Sed88]); this change would be simple to implement. Once the concept itself has been found, finding the state is just a matter of looking for the appropriate slot in the concept data structure.

In addition to having a concept network, each agent also has a discrimination tree which implements its state transition network. When the agent wishes to update its beliefs based on a particular move in the dialogue, it traverses the levels of the tree to find the new state at one of the leaves. Starting from the top node, the agent chooses the child labelled with the dialogue game of the move for which it is computing an update. It then chooses the child of that node corresponding to the type of move (initiation, response, or feedback), the child of that node corresponding to whether the self or the partner made the move, and the child of that node corresponding to the old state of the concept. All levels of the tree except the bottom one are implemented using two element list structures where the first element is the key and the second element is a list of the child nodes. We exploit the fact that the states are numbered sequentially and omit the keys at the bottom level, indexing the old state of the concept by position in the list. The leaf of the tree that is reached usually contains the number of the state which the concept is in after the agent hears the dialogue move. However, in the cases where the transition network has two arcs with the same label leaving the same state, the leaf node contains a code which triggers a procedure that the agent uses to decide between the two possibilities. In the case of the belief model implemented in the dialogue system, code U1 triggers a procedure that chooses state 6 if the agent "understands" the propositional content of the move (i.e., if all of the concepts involved are vivid for it) and state 3 otherwise. Once the new state is found using this structure, it is stored in the appropriate slot of the concept.

So far we have explained how the belief update for the central concept is computed; we still have to explain how belief propagation is implemented in line with the set of rules given in section 6.7. The most simple-minded implementation of the set of rules would be to apply the entire set of them in whatever order is convenient to the central concept of the dialogue move, keeping a list of the parents and children that change. Then it would suffice to work down the list recursively. This procedure will terminate as long as all rules only change the states of surrounding nodes so that either the dimension of the state corresponding to the self's beliefs or the dimension of the state corresponding to the partner's beliefs is the same as that of the node from which the change is propagated. For instance, this does not allow a rule such as "if a node becomes vivid to the partner, mark all parents as *not* vivid to the partner". This condition guarantees that for each node in the network, the state will never change more than a finite number of times.

The simulation doesn't use the most simple-minded implementation. Instead, the rules are coded into a three-phase procedure; first it calculates changes to the self's beliefs, then it calculates changes to the partner's beliefs, and finally it calculates changes to what the partner believes about the self's beliefs. First it computes a combination of rules 1 and 2 for the

self's beliefs working up through parents recursively until no changes occur. Then it works on the partner's beliefs. It computes a combination of rules 1 and 2 for the partner's beliefs in the same way as it did for the self. Then it computes rules 3, 4, and 5 for the central node, completing the computation of changes to the partner's beliefs. Then it computes a combination of rules 6 and 7 for the partner's beliefs about the self in the same way as it did for rules 1 and 2. Finally, it computes the results of rule 8. This procedure is more efficient than the simple-minded implementation, but it relies on the fact that the concept network is a strict AND/OR tree (i.e., that no node has a parent or child of the same type) and on the forms of these particular rules, especially the fact that each operates on only one of the nesting levels represented, to pick out exactly where recursion may occur.

D.2 How Queries about Beliefs are Handled

The belief model allows two kinds of queries: questions about how the agent thinks each of the agents stands with respect to particular concepts, and questions about the conceptual structure of the domain. Queries about the state of a concept are handled by an interface that decodes the meaning of the state numbers. This is implemented by using one of the two indices to the concept network (retrieving a concept by internal name or by referring expression) and by using one of a number of boolean functions which take a state and return particular information about what that state represents. For instance, there are boolean functions for "vivid for partner?", "vivid for self?", "uncertain about what the partner thinks?", and so on. The set of queries which is implemented is not comprehensive in that it does not cover all of the possible queries that one might wish to ask, but it suffices for the dialogue system and it would be trivial to extend it.

Quite often agents wish to know not just what state a concept is in, but what propositions it knows about a concept. Propositional beliefs are represented by the structure of the concept network; for example, the proposition "the swamp is to the left of the palm beach" (or *(describes swamp-0 (left-of palm-beach-0))*, as it might be represented in an agent's internal language) is represented by the fact that the swamp concept has as one of its children an AND description node, two of whose children are the concepts denoted by "left of" and "palm beach", and which has one more child which represents the propositional "glue" of how to put together these children to come up with the correct description. The system contains a routine which can take any sentence of the agent's internal language with any part of that sentence omitted in favour of a variable except the key word "describes", and will return a (possibly empty) list of all structures in the network that match the sentence (except that, for historic reasons,

it only returns those structures for which all the concepts are vivid). Thus if the agent asks itself (describes swamp-0 ?X) it will receive in return all of the descriptions which it believes hold about the swamp and which it also understands, if any. Only descriptions and questions about knowledge up to two nesting levels are allowed by this routine. This baroque structure should be replaced with a routine which allows queries about the structure of the network only, relying on the state interface to handle questions about the state of the concept (including the state of propositions, since they can be checked by looking at the "glue" node).

Bibliography

[AI86] Jose Antonio Ambros-Ingerson. Relationships between planning and execution. *AISB Quarterly*, (57), 1986.

[AK88a] Doug Appelt and Kurt Konolige. A nonmonotonic logic for reasoning about speech acts and belief revision. In *Second Workshop on Non-Monotonic Reasoning*, 1988.

[AK88b] Doug Appelt and Amichai Kronfeld. A descriptive model of reference using defaults. Technical Report TN 440, SRI, 1988.

[All83] James Allen. Recognising intentions from natural language utterances. In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*, pages 107–166. MIT Press, 1983.

[App85] Doug Appelt. *Planning English Sentences.* Cambridge U. Press, 1985.

[AS91] James F. Allen and Lenhart K. Schubert. The trains project. TRAINS Technical Note 91-1, University of Rochester, May 1991.

[Bal87] Afzal Ballim. The subjective ascription of belief to agents. In John Hallam and Chris Mellish, editors, *Advances in AI: Proceedings of AISB-87.* John Wiley, 1987.

[Bar86] Richard A. Bartle. Three ways to cross-level plan. Technical Report CSM-84, Essex, February 1986.

[BASY84] G. Brown, A. Anderson, R. C. Shillcock, and G. Yule. *Teaching Talk.* Cambridge University Press, 1984.

[Car83] Lauri Carlson. *Dialogue Games: An Approach to Discourse Analysis.* D. Reidel, 1983.

[Car90] Sandra Carberry. *Plan Recogntion in Natural Language Dialogue.* MIT Press, 1990.

[Caw89a] Alison Cawsey. *Generating Explanatory Discourse: A Plan-based, Interactive Approach.* PhD thesis, Edinburgh University, 1989.

[Caw89b] Alison Cawsey. The structure of tutorial discourse. Technical Report 425, DAI, University of Edinburgh, 1989.

[Cla87] William J. Clancey. *Knowledge-based Tutoring: The GUIDON Project.* MIT Press, 1987.

[Cla89] Andy Clark. *Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing.* MIT Press, 1989.

[CM81] Herbert H. Clark and Catherine R. Marshall. Definite reference and mutual knowledge. In Aravind K. Joshi, Bonnie L. Webber, and Ivan A. Sag, editors, *Elements of Discourse Understanding,* pages 10–63. Cambridge University Press, 1981.

[Coh81] Philip R. Cohen. The need for referent identification as a planned action. In *Proceedings of IJCAI-81,* pages 31–35, August 1981.

[CRMM87] Eugene Charniak, Christopher K. Riesbeck, Drew V. McDermott, and James R. Meehan. *Artificial Intelligence Programming.* Lawrence Erlbaum Associates, second edition, 1987.

[CS89] Herbert Clark and Edward Schaefer. Contributing to discourse. *Cognitive Science,* 13, 1989.

[Dal88] Robert Dale. *Generating referring expressions in a domain of objects and processes.* PhD thesis, Edinburgh University, 1988.

[DGH79] R. Duda, J. Gaschnig, and P. Hart. Model design in the prospector consultant system for mineral exploration. In D. Michie, editor, *Expert Systems in the Micro Electronic Age.* Edinburgh University Press, 1979.

[Dor85] Jim Doran. The computational approach to knowledge, communication and structure in multi-actor systems. In G. Nigel Gilbert and Christian Heath, editors, *Social Actions and Artificial Intelligence: Surrey Conferences on Sociological Theory and Method 3,* pages 160–171. Gower Publishing Company, 1985.

[FH88] Ronald Fagin and Joseph Y. Halpern. Belief, awareness, and limited reasoning. *Artificial Intelligence,* 34, 1988.

[FHN72] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence,* 3:251–288, 1972.

[FR86] Giacomo Ferrari and Ronan Reilly. A two-level dialogue representation. In *Proceedings of the Eleventh International Conference on Computational Linguistics*, 1986.

[Fri87] Alan Frisch. Inference without chaining. In *Proceedings of IJCAI-87*, pages 515–519, 1987.

[GH89] Michael Gerlach and Helmut Horacek. Dialog control in a natural language system. In *Proceedings of Cognitive Science 1989*, pages 27–34, 1989.

[GI89] Michael P. Georgeff and Francois Felix Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of IJCAI-89*, 1989.

[Gil87] G. Nigel Gilbert. Advice, discourse, and explanations. In *Proceedings of the Third Alvey Explanation Workshop*, September 1987.

[Goo86] Bradley A. Goodman. Reference identification and reference identification failures. *Computational Linguistics*, 12(4), 1986.

[Had89] Robert F. Hadley. The many uses of 'belief' in a.i. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 115–122, 1989.

[Hay75] Philip J. Hayes. A representation for robot plans. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, pages 181–188, 1975.

[HM85] Joseph Y. Halpern and Yoram Moses. A guide to the modal logics of knowledge and belief: Preliminary draft. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 480–490, 1985.

[Hou86] George Houghton. *The Production of Language in Dialogue: A Computational Model.* PhD thesis, University of Sussex, April 1986.

[Hov87] Eduard H. Hovy. *Generating Natural Language under Pragmatic Constraints.* PhD thesis, Yale University, 1987.

[HRHR79] Barbara Hayes-Roth and Frederick Hayes-Roth. A cognitive model of planning. *Cognitive Science*, 3(4):275–310, 1979.

[Isr85] David Israel. A weak logic of knowledge and belief: epistemic and doxastic logic for the yuppie generation. Technical Report TN 359, SRI, 1985.

[IYA90] Hitoshi Iida, Takayuki Yamaoka, and Hidekazu Arita. Three typed pragmatics for dialogue structure analysis. In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, volume 3, pages 370–372, 1990.

[KA86] Henry Kautz and James Allen. Generalized plan recognition. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 32–37, 1986.

[Kae87] Leslie Pack Kaelbling. An architecture for intelligence reactive systems. In Michael Georgeff and Amy Lansky, editors, *Reasoning about Actions and Plans*. Morgan Kaufmann, 1987.

[Kas88] Robert John Kass. Acquiring a model of the user's beliefs from a cooperative advisory dialog. Technical Report MS-CIS-88-104, LINC LAB 139, University of Pennsylvania, Department of Computer and Information Science, December 1988.

[KID91] Jacqueline C. Kowtko, Stephen D. Isard, and Gwyneth M. Doherty. Conversational games within dialogue. In *DANDI Workshop on Discourse Coherence*, 1991.

[Kon86] Kurt Konolige. *A Deduction Model of Belief*. Morgan-Kaufmann/Pitman, 1986.

[Kon87] Kurt Konolige. On the relation between default theories and autoepistemic logic. In *Proceedings of IJCAI-87*, pages 394–401, 1987.

[Kon88] Kurt Konolige. Hierarchic autoepistemic theories for nonmonotonic reasoning: Preliminary report. Technical Report TN 446, SRI, 1988.

[LA87] Diane Litman and James Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

[Lev83] Stephen C. Levinson. *Pragmatics*. Cambridge U. Press, 1983.

[Lev84] H. J. Levesque. A logic of implicit and explicit belief. In *Proceedings of AAAI-84*, 1984.

[LM77] James A. Levin and James A. Moore. Dialogue games: Metacommunication structures for natural language interaction. *Cognitive Science*, 1(4):395–420, 1977.

[Mar80] Mitchell P. Marcus. *A Theory of Syntactic Recognition for Natural Language*. MIT Press, 1980.

[MC88] Kathleen McCoy and J. Cheng. Focus of attention: constraining what can be said next. In *Proceedings of the 4th International Workshop on Natural Language Generation*, 1988.

[McA87] Karen McAllister. The correction of misunderstanding in man-machine dialogue. In *Proceedings of the Third Alvey Explanation Workshop*, September 1987.

[McK85] Kathleen R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, 1985.

[McT85] Michael McTear. Breakdown and repair in naturally occurring conversation and human-computer dialogue. In G. Nigel Gilbert and Christian Heath, editors, *Social Actions and Artificial Intelligence: Surrey Conferences on Sociological Theory and Method 3.* Gower Publishing Company Limited, 1985.

[Moo84] Robert C. Moore. Possible-world semantics for autoepistemic logic. Technical Note 337, SRI Artificial Intelligence Center, 1984.

[Moo90] Johanna Doris Moore. A reactive approach to explanation in expert and advice-giving systems. Special Report 251, University of Southern California/Information Sciences Institute, 1990.

[MRS82] Gordon I. McCalla, Larry Reid, and Peter F. Schneider. Plan creation, plan execution and knowledge acquisition in a dynamic microworld. *International Journal of Man-machine Studies*, 16:89–112, 1982.

[MS87] Stuart A. MacMillan and Derek H. Sleeman. An architecture for a self-improving instructional planner for intelligence tutoring systems. *Computational Intelligence*, 3:17–27, 1987.

[MT87] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: A theory of text organization. Reprint Series 190, ISI, 1987.

[PIB87] Martha E. Pollack, David J. Israel, and Michael E. Bratman. Toward an architecture for resource-bounded agents. Technical Note 425, SRI, July 1987.

[PM86] Darwyn Peachey and Gordon McCalla. Using planning techniques in intelligent tutoring systems. *International Journal of Man-Machine Studies*, 24, 1986.

[Pow74] Richard J. D. Power. *A Computer Model of Conversation.* PhD thesis, University of Edinburgh, 1974.

[Ram87] Allan Ramsay. Knowing that and knowing what. In John Hallam and Chris Mellish, editors, *Advances in AI: Proceedings of AISB-87.* John Wiley, 1987.

[Ram89] Lance A. Ramshaw. A metaplan for problem-solving discourse. In *Proceedings of the Fourth European ACL*, April 1989.

[RDng] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proceedings of COLING-92*, 1992 (forthcoming).

[Rei90] Ehud Reiter. Generating descriptions that exploit a user's domain knowledge. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation.* Academic Press, 1990.

[Sab90] Gerard Sabah. Caramel: A flexible model for interaction between cognitive processes underlying natural language understanding. In *Proceedings of the Thirteenth International Conference on Computational Linguistics (COLING-90)*, August 1990.

[Sac73] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. In *Proceedings of IJCAI-73*, pages 412–422, 1973.

[SC75] J. M. Sinclair and R. M. Coulthard. *Towards an Analysis of Discourse: The English used by teachers and pupils.* Oxford University Press, 1975.

[Sed88] Robert Sedgewick. *Algorithms.* Addison-Wesley Publishing Company, 1988.

[Sel88] John Self. Bypassing the intractable problem of student modelling. In *Intelligent Tutoring Systems*, 1988.

[SF83] Ethel Schuster and Timothy Finin. Understanding misunderstandings. MS-CIS 12, Computer Science Dept., University of Pennsylvania, 1983.

[Sha84] Nigel R. Shadbolt. *Constituting reference in natural language: the problem of referential opacity.* PhD thesis, Edinburgh University, 1984.

[SR90] Sam Steel and Han Reichgelt. Knowing how and finding out. In *Final Report of the Ninth Workshop of the UK Planning Special Interest Group*, 1990.

[SSJ78] Henry Sacks, Emanuel Schegloff, and Gail Jefferson. A simplest systematics for the organization of turn-taking for conversation. In J. N. Schenkein, editor, *Studies in Organization of Conversational Interaction*, volume 1. Academic Press, 1978.

[Ste80] Mark Jeffrey Stefik. *Planning with Constraints.* PhD thesis, Stanford University, Dept. of Computer Science, 1980.

[Ste81] Mark Jeffrey Stefik. Planning and meta-planning. *Artificial Intelligence*, 16, 1981.

[Ste91a] Keith Stenning. Distinguishing conceptual and empirical issues about mental models. In Y. Rogers, A. Rutherford, and P. Bibby, editors, *Models in the Mind.* Academic Press, 1991.

[Ste91b] Keith Stenning. The role of sequence information in representations underlying discourse memory. In *DANDI Workshop on Discourse Coherence*, April 1991.

[Tho77] Henry Thompson. Strategy and tactics: A model for language production. In *Papers from the Thirteenth Regional Meeting of the Chicago Linguistics Society*, 1977.

[Wil83] Robert Wilensky. *Planning and Understanding: A Computational Approach to Human Reasoning.* Addison-Wesley Publishing Company, 1983.

[Wil88] David E. Wilkins. *Practical Planning.* Morgan Kaufmann, 1988.

[YS87] Richard M. Young and Tony Simon. Planning in the context of human-computer interaction. In *Proceedings of HCI '87*, 1987.